



LIBRARY OF THE  
UNIVERSITY OF ILLINOIS  
AT URBANA-CHAMPAIGN

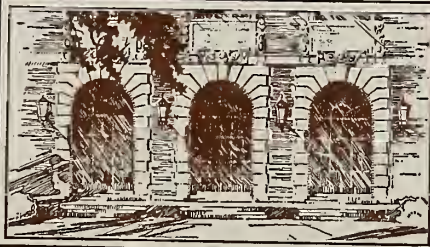
510.84

I l 6r

no. 713-714

cop. 2

[REDACTED] S







Digitized by the Internet Archive  
in 2013

<http://archive.org/details/programmanualnor714lah>







010101  
ILGM  
no 714  
Cop 2

Math

UIUCDCS-R-75-714

PROGRAM MANUAL:  
NOR NETWORK TRANSDUCTION BY GENERALIZED GATE MERGING  
AND SUBSTITUTION  
(Reference Manual of NOR Network Transduction  
Programs NETTRA-G3 and NETTRA-G4)

by

April 1975

H.C. Lai



THE LIBRARY OF THE

MAY 7 1975

UNIVERSITY OF ILLINOIS

DEPARTMENT OF COMPUTER SCIENCE  
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS





PROGRAM MANUAL:  
NOR NETWORK TRANSDUCTION BY GENERALIZED GATE MERGING  
AND SUBSTITUTION  
(Reference Manual of NOR Network Transduction  
Programs NETTRA-G3 and NETTRA-G4)

by

H.C. Lai

April 1975

Department of Computer Science  
University of Illinois at Urbana-Champaign  
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.



## ABSTRACT

This paper explains the operation and usage of two FORTRAN programs NETTRA-G3 and NETTRA-G4, developed for NOR-network transduction (transformation and reduction).

Existing (non-optimal) NOR-gate networks and their required output functions are given to the programs as input. Program NETTRA-G3 reduces the number of gates in the network by means of merging gates. When two gates are merged to one, the merged gate will be connected to the immediate successors of the two gates, and the two gates will be removed from the network. On the other hand, program NETTRA-G4 reduces the number of gates by means of substituting for all output corrections of a particular gate under consideration. If these connections can be successfully replaced, this particular gate then can be removed from the network.

These programs are only two out of a whole system of programs, designated by the name "NETTRA" (for NETWORK TRANSDUCTION), which implement different NOR-network transduction procedures.

The theoretical basis for the algorithms implemented by NETTRA-G3 and -G4 is detailed in earlier reports ([1] and [3]).



## ACKNOWLEDGMENT

The author is greatly appreciative of Professor S. Muroga for his discussions and guidance relating to the preparation of this paper, and also, for his careful reading and valuable suggestions for the improvement of the original manuscript. The author is also indebted to J. N. Culliney and Y. Kambayashi upon whose related research much of the work reported herein depends.

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.



## TABLE OF CONTENTS

	Page
1. INTRODUCTION . . . . .	1
2. A NETWORK TRANSDUCTION PROCEDURE BY MERGING OF GATES . . . . .	3
2.1 Procedure and Flowchart . . . . .	5
2.2 Examples. . . . .	9
3. A NETWORK TRANSDUCTION PROCEDURE BY SUBSTITUTION OF GATES. . . . .	15
3.1 Procedure and Flowchart . . . . .	17
3.2 Examples. . . . .	26
4. MAJOR FUNCTIONS OF COMMON SUBROUTINES. . . . .	34
5. INPUT DATA SETUP . . . . .	37
5.1 Input Data Card Format. . . . .	41
5.2 Restrictions on Problem Size. . . . .	52
5.3 Examples of Input Data Setup. . . . .	52
REFERENCES . . . . .	65
APPENDIX: PROGRAM LISTING . . . . .	66





## 1. INTRODUCTION

This manual is intended to instruct the reader in the use of the FORTRAN programs NETTRA-G3 and NETTRA-G4. These programs realize the algorithms described in detail in [1].

NETTRA-G3 and -G4 are only two out of the whole system of programs developed by the research group of Professor S. Muroga at the University of Illinois. The generic name 'NETTRA' (for NETWORK TRANSDUC-tion) designates the whole collection of programs comprising the system. All of the programs in the NETTRA system either transform or assist in transforming and reducing a large, non-optimal network of NOR gates realizing one or more functions into a smaller, less expensive (in terms of the number of required gates and connections, for example), near-optimal network realizing the same function(s). In general, such a transduction could involve a complete reorganization of the network: the addition and/or deletion of gates; the addition and/or deletion of connections among gates; and/or the substitution of certain connections for others. The transduction procedures realized by NETTRA-G3 and -G4 actually involve these alterations with the exception of adding gates to the network.

The procedures realized in these programs NETTRA-G3 and -G4 aim at reducing the number of NOR gates in the network by merging gates and substituting for gates, respectively. They are more complex than those appearing in programs NETTRA-PG1, -P1 and -P2 and require more computer time to execute. However, they are more powerful also, and they can often reduce a network when it is impossible to do so by the procedures in

NETTRA-PG1, -P1, and -P2. The programs NETTRA-PG1, -P1, and -P2 described in [2], though, are more efficient than NETTRA-G3 and -G4 when first applied to large, far from optimal networks. NETTRA-G3 and -G4 are most useful when applied to more nearly optimal networks which is fairly difficult to be further reduced.

The next two sections, Sections 2 and 3; will discuss the two programs in great detail and present some examples of the effectiveness of their transductions. This will be followed, in Section 4, by a description of the functions of subroutines which support the subroutines actually realizing the procedures. Section 5 outlines the preparation of input for these programs. Finally, in the Appendix, a listing of all of the FORTRAN programs NETTRA-G3 and NETTRA-G4 will be given.

## 2. A NETWORK TRANSDUCTION PROCEDURE BY MERGING OF GATES

This section will discuss the NOR-network transduction procedure realized by the FORTRAN program designated NETTRA-G3. This program realizes a procedure which merges two NOR gates at a time. In a given network, if two gates can be replaced by one gate with inputs from external variables and/or existing gates not fed by the two gates, they are said to be mergeable. The gate replacing those two gates is called the merged gate. The procedure realized by program NETTRA-G3 is the procedure to examine every pair of gates in a network to see if they are mergeable. For the sake of efficiency this procedure does not use the necessary and sufficient conditions for finding mergeable gates. Instead, the concept of compatible sets of permissible functions<sup>†</sup> (CSPF) is used.

The input to this program is a description of a particular NOR network under consideration. This description (explained in great detail in Section 5) consists of a set of various network parameters. The output of this program is a description of the "transduced" network (if a transduction, in this case the merging of gates, was possible).

The entire NETTRA-G3 program requires 127K bytes of core storage, about 42K bytes being occupied by the actual program instructions and about 85K by the stored data.

The gate merging procedure is realized by the FORTRAN subroutine GTMERG. This subroutine and the following support subroutines, written in FORTRAN IV for the IBM 360/75, constitute the program NETTRA-G3:

---

<sup>†</sup>The reader is assumed to be familiar with the definitions presented in [1], [3].

MAIN, GTMERG, MINI2, OUTPUT and SUBNET. Two system-supplied timing subroutines STIMZE and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without changing the procedure itself. The functions of the support subroutines MAIN, OUTPUT, and SUBNET will be discussed in Section 4. The function of subroutine MINI2 is to calculate compatible sets of permissible functions for all the gates in a given network. It is explained in great detail in [4] and [2].

The general organization of the program NETTRA-G3 is shown in Fig. 2.1. An arrow from block i to block j represents the fact that the subroutine represented by block i calls the subroutine represented by block j.

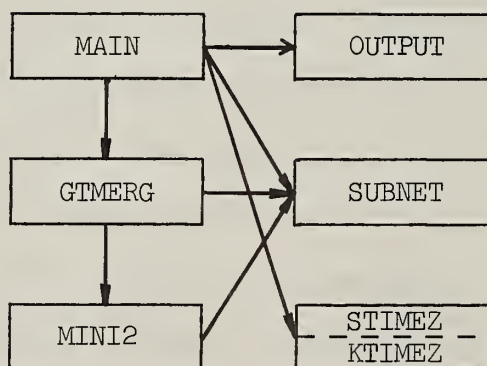


Fig. 2.1 General organization of the program NETTRA-G3.

Section 2.1 will describe the procedure itself, and Section 2.2 will show examples obtained by NETTRA-G3.

## 2.1 Gate Merging Procedure and Flowchart

The gate merging procedure is essentially performed by the subroutine GTMERG. GTMERG is programmed based on the procedure GMGC (a Generalized Merging of Gates using CSPF's) described in detail in [1]. The following discussions of GTMERG will assume knowledge of the information contained in [1].

The purposes of the main variables and arrays appearing in the subroutine will be explained in the program listing in the Appendix.

### PROCEDURE FOR MERGING OF GATES

#### Step 1. Calculation of CSPF's

Call subroutine MINI2 to calculate a compatible set of permissible functions for every gate in the network.

#### Step 2. Select a Pair of Gates

Select two gates GI and GJ such that  $GJ > GI$  both according to the ascending order of gate labels.

If all possible combinations of gate pairs have been considered, return to the calling subroutine.

#### Step 3. Calculation of CSPF for the Merged Gate

Calculate the CSPF for the merged gate GIJ by forming the intersection of the two CSPF's for gates GI and GJ. If the CSPF for GIJ is a null set, GI and GJ are not mergeable, and go to Step 2.

Calculate the set of connectable functions for the merged gate GIJ.



#### Step 4. Check Substitutability

If the output of gate GI is included in the CSPF of gate GJ and gate GI is not a successor of gate GJ, then gate GJ can be replaced by gate GI, and go to step 8.

If the output of gate GJ is contained in the CSPF of gate GI, and gate GJ is not a successor of gate GI, then gate GI can be replaced by gate GJ. So interchange the labels of gates GI and GJ and go to step 8.

#### Step 5. Select Connectable Functions for Gate GIJ

Find all external variables and/or existing gates in the network which are contained in the set of connectable functions, and which are not fed by gate GI or GJ.

#### Step 6. Check Realizability of the Merged Gate

If connectable functions obtained in Step 5 do not realize a function contained in the CSPF of GIJ, go to step 2.

#### Step 7. Construct the Merged Gate

Disconnect all input connections of gate GI, connect all connectable functions found in Step 5 to gate GI to realize the merged gate at gate GI.

#### Step 8. Substitution

Disconnect all output connections of gate GJ, and connect gate GI to all immediate successors of gate GJ. The resulting network still realizes the specified function(s).



Step 9. Update Information on the New Network

Call subroutine SUBNET to update the information on the configuration of the network (e.g., predecessor lists and successor lists).

Go to Step 1.

The flowchart of this procedure, realized by FORTRAN subroutine GTMER is shown in Fig. 2.2.

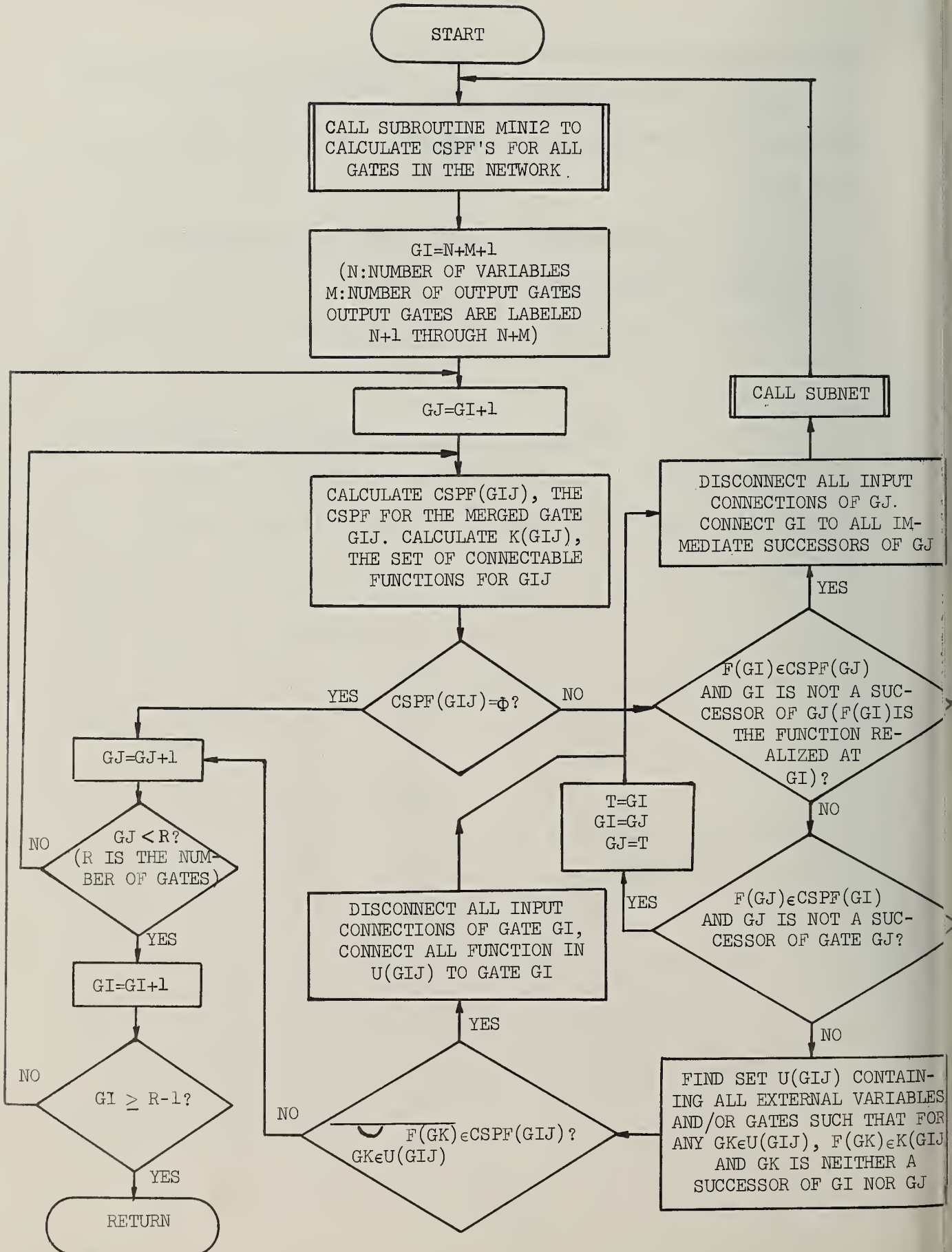


Fig. 2.2 Flowchart of subroutine GTMERG.

## 2.2 Examples for Program NETTRA-G3

The printout obtained during the solution of a typical problem by NETTRA-G3 is shown in Fig. 2.3. The original network, as specified in the beginning of the printout (Fig. 2.3 (a)), consists of 33 gates and 310 connections and realizes a single 5-variable output function. Only uncomplemented variables are assumed to be available as inputs to the network.

This information is followed by a complete truth table (b) showing the output of every gate in the original network for every possible input combinations. Note that it is gate 1 which realizes the output function of the network.

Next appears a description of the configuration of the network (c). Each gate is listed along with the gates and/or external variables which are its inputs. The level numbers, also to be seen in (c), will be discussed in Section 5.3.

The truth table (note that the outputs for disconnected gates are shown as all 1's) and network configuration for the transduced network resulting from the execution of NETTRA-G3 are shown in (d) and (e), respectively. The derived network, obtained in .74 seconds, consists of 12 gates and 38 connections.

\*\*\*\* 5 VAR. EXAMPLE

HEX=8B5809F0

NUMBER OF VARIABLES = 5

NUMBER OF FUNCTIONS = 1

COST COEFFICIENT A =1000

B = 1

--- UNCOMPLEMENTED VARIABLES X ---

ORIGINAL NETWORK COST=33310

(a) Heading information and network parameter

Fig. 2.3 Printout obtained from NETTRA-G3 for a sample problem.

## TRUTH TABLE

```

X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0
 2 = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 3 = 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 4 = 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 5 = 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 6 = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 7 = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 8 = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 9 = 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 = 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12 = 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13 = 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14 = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
18 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
19 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
20 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
22 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
23 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
24 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
25 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
26 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
27 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
28 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
29 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
30 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
31 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
32 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
33 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

```

(b) Truth table for original network.

GATE ..	LEVEL	FED BY
1	/ 1/	3 4 5 7 10 12 15 16 17 18 19 20 21 23 24 30 31 32 33
2	/ 7/	X1 X2 X3 X4 X5
3	/ 6/	X1 X2 X3 X4 2
4	/ 6/	X1 X2 X3 X5 2
5	/ 5/	X1 X2 X3 2 3 4
6	/ 6/	X1 X2 X4 X5 2
7	/ 5/	X1 X2 X4 2 3 6
8	/ 5/	X1 X2 X5 2 4 6
9	/ 4/	X1 X2 2 3 4 5 6 7 8
10	/ 6/	X1 X3 X4 X5 2
11	/ 5/	X1 X3 X4 2 3 10
12	/ 5/	X1 X3 X5 2 4 10
13	/ 4/	X1 X3 2 3 4 5 10 11 12
14	/ 5/	X1 X4 X5 2 6 10
15	/ 4/	X1 X4 2 3 6 7 10 11 14
16	/ 4/	X1 X5 2 4 6 8 10 12 14
17	/ 3/	X1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
18	/ 6/	X2 X3 X4 X5 2
19	/ 5/	X2 X3 X4 2 3 18
20	/ 5/	X2 X3 X5 2 4 18
21	/ 4/	X2 X3 2 3 4 5 18 19 20
22	/ 5/	X2 X4 X5 2 6 18
23	/ 4/	X2 X4 2 3 6 7 18 19 22
24	/ 4/	X2 X5 2 4 6 8 18 20 22
25	/ 3/	X2 2 3 4 5 6 7 8 9 18 19 20 21 22 23 24
26	/ 5/	X3 X4 X5 2 10 18
27	/ 4/	X3 X4 2 3 10 11 18 19 26
28	/ 4/	X3 X5 2 4 10 12 18 20 26
29	/ 3/	X3 2 3 4 5 10 11 12 13 18 19 20 21 26 27 28
30	/ 4/	X4 X5 2 6 10 14 18 22 26
31	/ 3/	X4 2 3 6 7 10 11 14 15 18 19 22 23 26 27 30
32	/ 3/	X5 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
33	/ 2/	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

(c) Configuration of original network.



[illegible]

(d) Truth table for transduced network



GATE ..	LEVEL	FED BY
1	/ 1/	12 21 31 32 33
2	/ 3/	X1 X2 X4 X5
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 3/	X1 X2
9	/ 1/	
10	/ 1/	
11	/ 1/	
12	/ 2/	X1 X3 X5 2
13	/ 1/	
14	/ 3/	X1 X4 X5
15	/ 1/	
16	/ 1/	
17	/ 1/	
18	/ 1/	
19	/ 1/	
20	/ 1/	
21	/ 2/	X2 X3 2
22	/ 3/	X2 X4 X5
23	/ 1/	
24	/ 1/	
25	/ 3/	X2
26	/ 1/	
27	/ 3/	X3
28	/ 1/	
29	/ 1/	
30	/ 1/	
31	/ 2/	X4 14 22 27
32	/ 2/	X5 8 14 22 27
33	/ 2/	14 25 27

\* A NETWORK DERIVED BY GTMERG  
COST = 12038

(e) Configuration of transduced network.

### 3. A NETWORK TRANSDUCTION PROCEDURE BY SUBSTITUTING FOR GATES

This section will discuss the NOR-network transduction procedure realized by the FORTRAN program designated NETTRA-G<sup>4</sup>. This procedure examines every output connection of a selected gate to see if it can be substituted for by connections from external variables and/or existing gates. The concept of compatible sets of permissible functions and a possibly connectable condition<sup>†</sup> are used in this procedure in searching the candidates of substitution. The substitutions will be performed only when all the output connections of the selected gate can be replaced. Therefore, any possible transduction performed by this procedure will reduce the number of gates in the network.

The input to this program is a description of a particular NOR-network under consideration. This description (explained in great detail in Section 5) consists of a set of various network parameters. The output of this program is the description of the "transduced" network (if a transduction, in this case the substitution of gates, was possible).

The entire NETTRA-G<sup>4</sup> program requires 146K bytes of core storage, about 56K bytes being occupied by the actual program instructions and about 90K bytes by the stored data.

The gate substitution procedure is realized by the FORTRAN subroutine PROCV. This subroutine along with the following support subroutines, written in FORTRAN IV for the IBM 360/75, constitutes the program NETTRA-G<sup>4</sup>: MAIN, PROCV, RQRNW, OUTPUT and SUBNET. Two system-supplied timing subroutines STIMEZ and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or

---

<sup>†</sup>The reader is assumed to be familiar with the definitions presented in [1] and [3].

another suitable timing routine substituted, without changing the procedure itself. The functions of the support subroutines MAIN, OUTPUT, SUBNET will be discussed in Section 4.

The general organization of the program NETTRA-G<sup>4</sup> is shown in Fig. 3.1. An arrow from block i to block j represents the fact that the subroutine represented by block i calls the subroutine represented by block j.

Section 3.1 will discuss the procedure itself, and Section 3.2 will show examples obtained by NETTRA-G<sup>4</sup>.

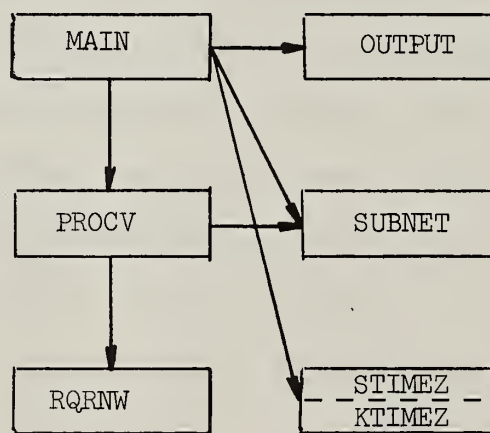


Fig. 3.1 General organization of the program NETTRA-G<sup>4</sup>.

### 3.1 Gate Substitution Procedure and Flowchart

The gate substitution procedure is essentially performed by the subroutine PROCV. Subroutine RQRNW, which is called by subroutine PROCV, calculates a compatible set of permissible functions for every output connection of a particular selected gate and for every gate in the network except that particular gate. For the sake of efficiency, this subroutine uses an ordering which is especially designed to fit the purpose of substituting for the selected gate. For this reason, the general subroutine for calculating CSPF's, FORTRAN subroutine MINI2 used in program NETTRA-G3 and some other NETTRA programs, is not used in NETTRA-G4.

Subroutine PROCV is programmed based on the procedure SOGPC described in [1]. The following discussions of PROCV will assume knowledge of the information contained in [1].

The purposes of the main variables and arrays appearing in the subroutine will be explained in the program listing in the Appendix.

## PROCEDURE FOR SUBSTITUTION OF GATES

### Step 0. Initialize

### Step 1. Selection of Gates

According to the ascending order of gate labels, select a gate GI satisfying the following conditions:

1. GI is not an output gate of the network,
2. GI has no single output gate as its immediate predecessor.

If all gates have been selected, return to the calling subroutine (MAIN in NETTRA-G4).

### Step 2. Remove Internal Inverters

If GI is the only immediate predecessor of a gate GJ not being an output gate, then GJ is redundant. Remove GJ from the network, and connect all immediate predecessors of gate GI to all immediate successor of gate GJ. Call subroutine SUBNET to update the information on the configuration of the network (successor lists, predecessor lists, etc.), and then go to Step 1.

### Step 3. Calculation of CSPF's

Call subroutine RQRNW to calculate CSPF's for all output connections of gate GI, and for all gates except gate GI.

### Step 4. Selection of Connections

Select one output connection of gate GI. Let the connection be GIGJ (indicating the connection from gate GI to gate GJ). The connection is selected according to the ascending order of GJ.

If all connections from GI have been considered, go to step 8.

### Step 5. Partition of Candidates for Substitution

Partition external variables and/or existing gates which are not successors of gate GI into the following subsets.

- (1) Effectively-connectable functions which satisfy the following conditions
  - a. must have 0-components corresponding to all 0-components in the CSPF of GIGJ.
  - b. must have at least one 1-component corresponding to a 1-component in the CSPF of GIGJ.
- (2) Possibly-connectable functions which may actually have 1-components corresponding to some 0-components in the CSPF of GIGJ, but the corresponding components in the CSPF's of the functions themselves must be \*. The second condition in (1) is also required for a function being in this subset.
- (3) All other functions.

#### Step 6. Selection of Connectable Functions

From connectable functions, select functions which are essential for replacing connection GIGJ.

If these functions contain at least one 1-component corresponding to every 1-component in CSPF of GIGJ, connect these functions to gate GI and call RQRGT (an entry point in subroutine RQRNW) to update CSPF's for all gates feeding these functions, and go to Step 4.

#### Step 7. Selection of Possibly-Connectable Functions

Select possibly-connectable functions to replace connection GIGJ according to the following substeps.

7-1 Select one possibly-connectable function GK which has at least one 1-component corresponding to a 1-component in the CSPF of GIGJ which is not covered by any other functions having been selected.

If all possibly-connectable functions have been considered, then connection GIGJ is not replaceable, and go to Step 9.



- 7-2 Find a function GL satisfying that (1) GL is not a successor of GI, (2) connecting GL to GK would make GK a connectable function to GJ without changing the essential 1-components in GK. If no such GL can be found, go to Step 7-1.
- 7-3 Connect GL to GK and GK to GJ. Call SUBNET to update the information on the network (successor lists etc.), call RQRGT to update the CSPF's for gates feeding GK, and call UPTRTH (another entry point in RQRNW) to update the truth table for gates which are successors of GK.
- 7-4 If all 1-components in the CSPF of GIGJ are covered by the selected functions, go to step 4; otherwise go to Step 7-1.

#### Step 8. Removing Gate

Disconnect all connections going to or coming out from gate GI. Call SUBNET to update the information on the network. Go to Step 1.

#### Step 9. Restoring the Network

Since GIGJ could not be replaced, restore the old network (the network before gate GJ was taken into consideration). Go back to Step 1.

According to the procedure described above, the gates which could not be replaced when they were chosen would not be considered again although they may become replacable after some other gate has been removed. The user can easily change the subroutine MAIN to repeatedly apply this procedure until no further improvement could be found. Another way to do this is simply to change the statement 'go to Step 1' in Step 8 to ' go to Step

The flowchart of subroutine PROCV is shown in Fig. 3.2.

The calculation of CSPF's for gates and connections from the selected gate is accomplished by subroutine RQRNW which is called in



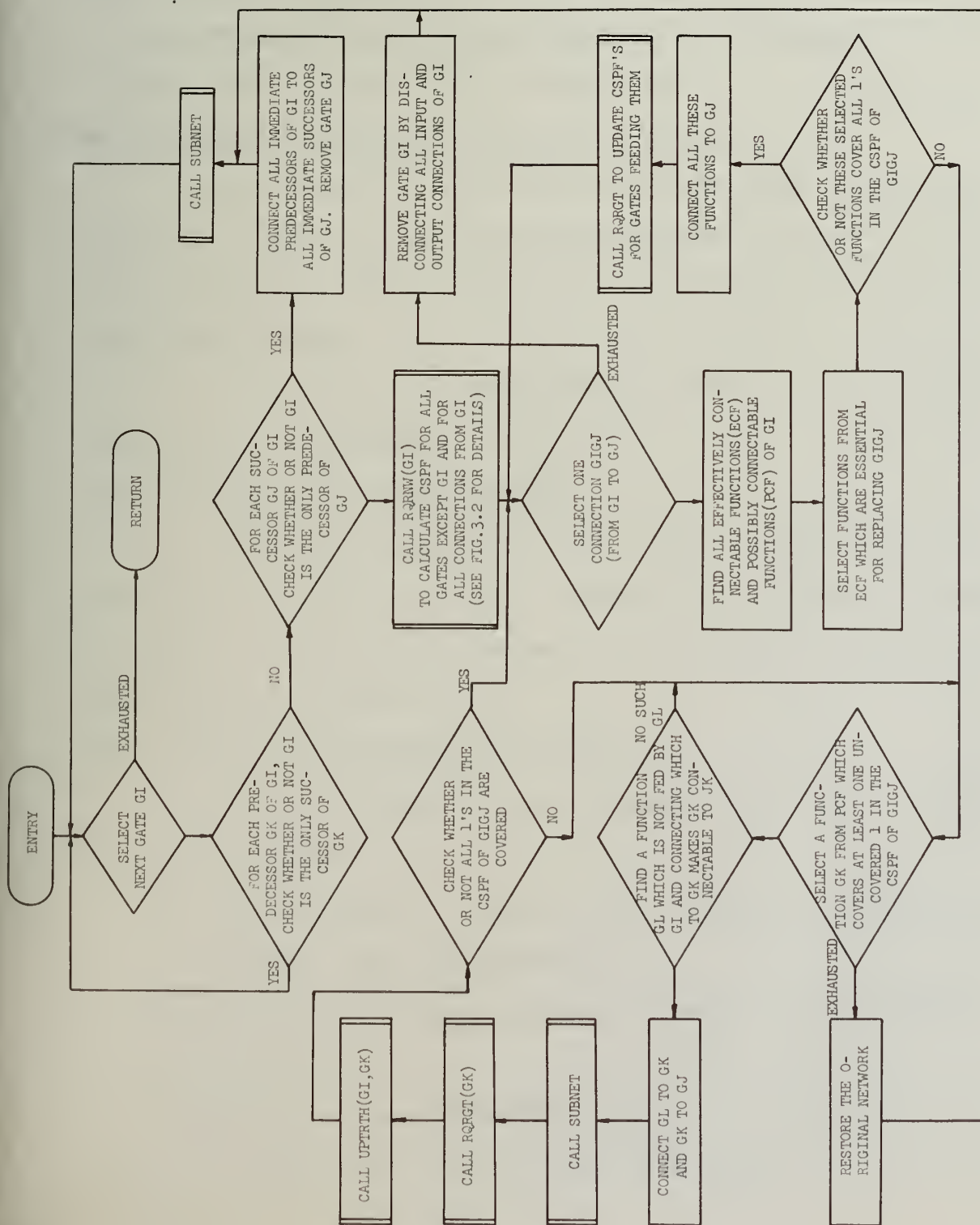


Fig. 3.2 Flowchart of subroutine PROCV.

step 3 of the above procedure. The CSPF for a gate is calculated by forming intersections of CSPF's for all its output connections.

Since the CSPF's for connections (except the connections from the selected gate) need not be stored in the program, only the CSPF for each gate is stored. For the details of the process of the calculation of CSPF's, see [1]. The following is the procedure realized by subroutine RQRNW.

Note that gate GI is specified by the calling subroutine (PROCV) when RQRNW is called.

## PROCEDURE FOR CALCULATION OF CSPF's

### Step 0. Initialization

Assign \* to every component of the CSPF for every gate in the network.

### Step 1. CSPF for Output Gates

Assign CSPF's of the output gates in the network the corresponding output functions specified by the input data.

### Step 2. Selection of Gates

Select a gate GJ other than GI according to the ordering of gate levels, i.e., gates in a lower level are chosen prior to gates in a higher level, and gates in the same level are chosen according to the ascending order of gate labels.

If all gates have been considered, return to the calling subroutine (PROCV in NETTRA-G4.)

### Step 3. Selection of Input Connection

Select an input connection GKGJ of gate GJ, (GKGJ denotes the connection from gate GK to gate GJ). The selection is based on the ascending order of GK with an exception that GIGJ (if exists) is selected last among all input connections to GJ.

### Step 4. Calculations of CSPF's for Connections

Calculate the CSPF for input connection GKGJ of gate GJ according to the following rules:

- (1) For 1-components in the CSPF of GJ, the corresponding components of CSPF for GKGJ are assigned 0.
- (2) For 0-components in the CSPF of GJ, there are the following three cases.
  - (a) Each of the corresponding components in the CSPF of GKGJ is assigned \*, if the corresponding component of the function

of GK is 0.

- (b) Each of the corresponding components in the CSPF of GKGJ is assigned 1, if the corresponding component of the function of GK is 1, and no corresponding components of other input connections have been assigned 1 before GKGJ is taken into consideration.
- (c) All other components in the CSPF of GKGJ are assigned \*'s.
- (d) For \*-components in the CSPF of GJ, the corresponding components of CSPF of the CSPF of GKGJ are assigned \*'s.

#### Step 5. Calculation of CSPF for Gate<sup>†</sup>

If  $GK \neq GI$ , take intersection of the CSPF of GKGJ with the intermediate CSPF obtained so far for gate GJ, (i.e., the intersection of CSPF's for input connections of GJ selected prior to GKGJ). This intersection is the new intermediate CSPF for gate GJ (the final CSPF for GJ if GKGJ is selected the last among all input connections to GJ).

Go to Step 3.

The flowchart of this subroutine is shown in Fig. 3.3.

When RQRNW is entered through the entry point RQRGT(GZ), the procedure is applied only to predecessors of gate GZ. When RQRNW is entered through the entry point UPTRTH(GI,GZ), the procedure is applied only to gates which are successors of gate GZ but not gate GI.

---

<sup>†</sup>This step is actually performed while each component of CSPF for GKGJ is assigned.

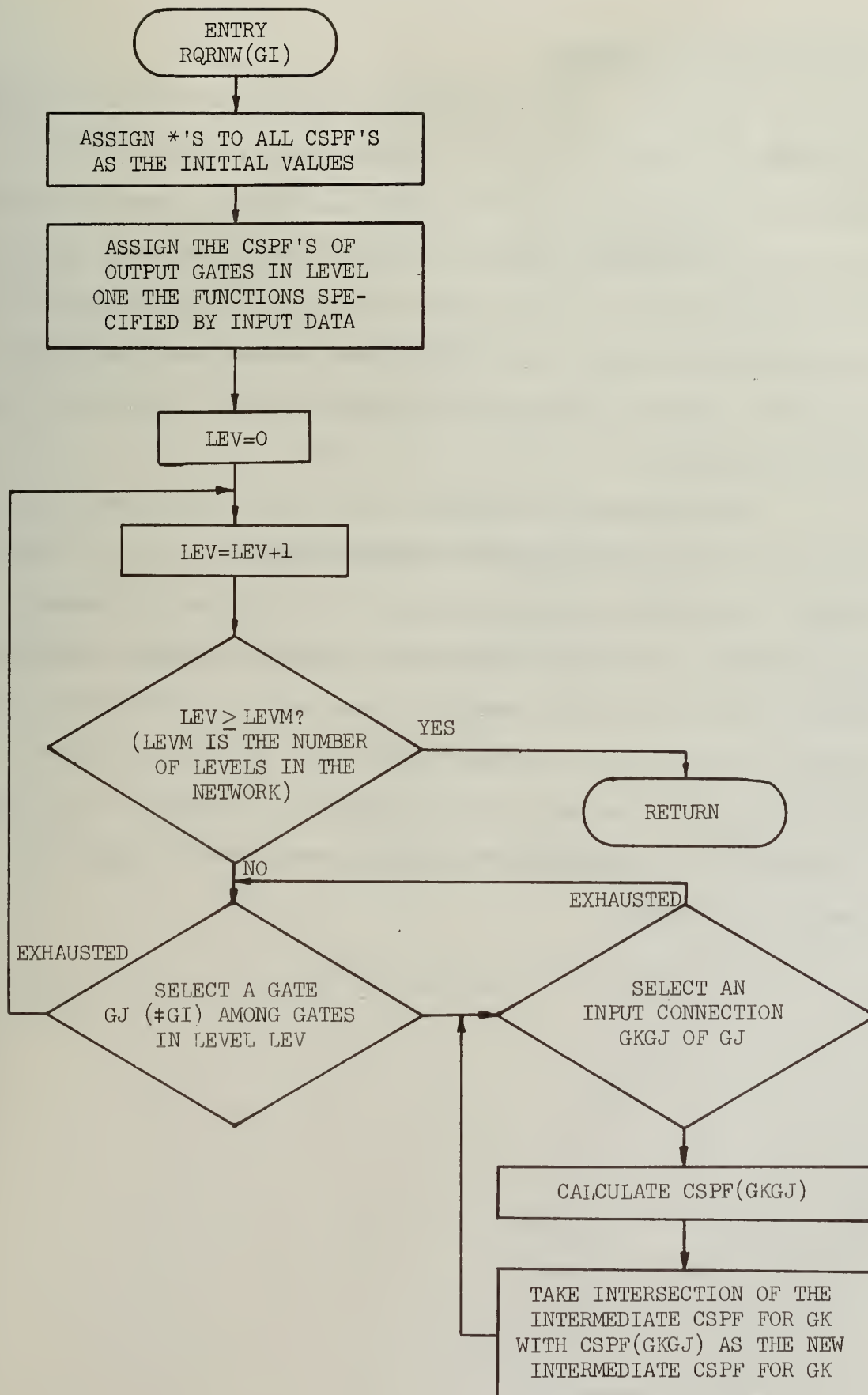


Fig. 3.3 Flowchart of subroutine RQRNW.

### 3.2 Examples for NETTRA-G4

The printout obtained during the solution of a typical program by NETTRA-G4 is shown in Fig. 3.5. The printout is similar to the one obtained by NETTRA-G3 discussed in Section 2.2 with (a) showing the heading information and parameters for that problem, (b) the truth table of the original network, (c) the configuration of the original network obtained from input cards, (d) the truth table of the transduced network obtained by applying subroutine PROCV, and (e) the configuration of that transduced network.

The procedure for merging of gates and the procedure for substitution of gates can be used in a combined program. Fig. 3.6 shows an example obtained by a combined program which is similar to program NETTRA-G3 with an additional statement 'CALL PROCV' inserted immediately after the statement 'CALL GTMERG' in subroutine MAIN of NETTRA-G3. The original network is the one shown in Fig. 2.3. It can be seen that PROCV improved the network obtained by NETTRA-G3 by reducing 1 gate and 4 connections. However, this does not mean that PROCV is always more powerful than GTMERG.



\*\*\*\*\* 5 VAR. EXAMPLE

HEX=3B5809F0

NUMBER OF VARIABLES = 5

NUMBER OF FUNCTIONS = 1

COST COEFFICIENT A = 1000

B = 1

--- UNCOMPLEMENTED VARIABLES X ---

ORIGINAL NETWORK COST= 21086

(a) Heading information and network parameters.

Fig. 3.5 Printout obtained from NETTRA-G4 for a sample problem.



## TRUTH TABLE

X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 X2 = 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1  
 X3 = 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1  
 X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1  
 X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1  
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0  
 2 = 1 0  
 3 = 0 1 0  
 4 = 0 0 1 0  
 5 = 0 1 1 1 0  
 6 = 1 0 0 0 1 0  
 7 = 0 1 0 0 0 1 0  
 8 = 0 0 0 0 0 0 0 0 0 1 0  
 9 = 0 0 1 0 0 0 0 0 0 1 0 1 0  
 10 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 11 = 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0  
 12 = 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0  
 13 = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0  
 14 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
 15 = 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
 16 = 1 0 1 0 1 0 1 0  
 17 = 0 1 0 0 0 1 0 0 0 0 0 0  
 18 = 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0  
 19 = 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0  
 20 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0  
 21 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

(b) Truth table for original network.

GATE .. LEVEL	FED BY
1        / 1/	3 4 5 7 8 9 10 11 12 13 15 17 21
2        / 3/	X1 X2 X3 X4 X5
3        / 2/	X1 X2 X3 X4 2
4        / 2/	X1 X2 X3 X5 2
5        / 2/	X1 X2 X3 2
6        / 3/	X1 X2 X4 X5
7        / 2/	X1 X2 X4 6
8        / 2/	X1 X3 X4 X5 2
9        / 2/	X1 X3 X5 2
10       / 2/	X2 X3 X4 X5 2
11       / 2/	X2 X3 X4 2
12       / 2/	X2 X3 X5 2
13       / 2/	X2 X3 2
14       / 3/	X2 X4 X5
15       / 2/	X2 X4 14
16       / 3/	X1 X2 X5
17       / 2/	X2 X5 14 16
18       / 3/	X2
19       / 3/	X3
20       / 3/	X1 X4 X5
21       / 2/	18 19 20

(c) Configuration of original network.

TRUTH TABLE

```

X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0
 2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 8 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 9 = 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
12 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
13 = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
14 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
15 = 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
16 = 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
18 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
19 = 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
20 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1

```

(d) Truth table for transduced network.

GATE ..	LEVEL	FED BY
1	/ 1/	9 10 13 15 17 21
2	/ 1/	
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 1/	
9	/ 2/	X1 X3 X5 14
10	/ 2/	X2 X3 X4 X5 16
11	/ 1/	
12	/ 1/	
13	/ 2/	X2 X3 14
14	/ 3/	X2 X4 X5
15	/ 2/	X2 X4 14
16	/ 3/	X1 X2 X5
17	/ 2/	X2 X5 14 15
18	/ 3/	X2
19	/ 3/	X3
20	/ 3/	X1 X4 X5
21	/ 2/	18 19 20

\* A NETWORK DERIVED BY PROCV  
COST= 1239.

(e) Configuration of transduced network

TRUTH TABLE

```

X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0
 2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 8 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 9 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
12 = 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
14 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
17 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
18 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
20 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
21 = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
22 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
23 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
24 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
25 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
26 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
27 = 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0
28 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
29 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
30 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
31 = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0
32 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0
33 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

```

(a) Truth table of transduced network

Fig. 3. Printout obtained from a program applying GTMERG and PROCV for the same problem as that shown in Fig. 2.3.

GATE ..	LEVEL	FED BY
1	/ 1/	12 21 31 32 33
2	/ 1/	
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 3/	X1 X2
9	/ 1/	
10	/ 1/	
11	/ 1/	
12	/ 2/	X1 X3 X5 22
13	/ 1/	
14	/ 3/	X1 X4 X5
15	/ 1/	
16	/ 1/	
17	/ 1/	
18	/ 1/	
19	/ 1/	
20	/ 1/	
21	/ 2/	X2 X3 14
22	/ 3/	X2 X4 X5
23	/ 1/	
24	/ 1/	
25	/ 3/	X2
26	/ 1/	
27	/ 3/	X3
28	/ 1/	
29	/ 1/	
30	/ 1/	
31	/ 2/	X4 14 22 27
32	/ 2/	X5 8 14 22 27
33	/ 2/	14 25 27

\* A NETWORK DERIVED BY GTMERG AND PROCV  
COST= 11034

(b) Configuration of transduced network by GTMERG and PROCV.



#### 4. MAJOR FUNCTIONS OF COMMON SUBROUTINES

The subroutines realizing the procedures presented in the two previous sections share the support of three subroutines whose main functions will be discussed in this section: MAIN, SUBNET, OUTPUT. In addition, subroutine MINI2 described in [2] will also be briefly explained which is used in NETTRA-G3.

Complete program listings of these four subroutines can be found in the Appendix along with the listings of the subroutines realizing the previously described procedures.

The functions of these subroutines are as follows:

MAIN: This subroutine repeatedly reads in groups of input data which include information about the given networks; e.g., the number of external variables, whether or not the compliments of variables are also available as input variables, the number of output functions, the number of NOR gates, the list of connections, and the truth table of the output functions (see Section 5 for details). Using this information, MAIN constructs the incidence matrix, INC\$MX, for the network. INC\$MX is a two-dimensional array whose arguments represent gates or external variables. An array element  $\text{INC\$MX}(A1, A2) \geq 1$  indicates a connection from A1 to A2; an array element  $\text{INC\$MX}(A1, A2) \leq 0$  indicates the absence of a connection from A1 to A2. Next subroutine SUBNET is called to calculate the level of each gate and to make lists of predecessors and successors (i.e., which gates precede which and which gates succeed which). MAIN then prints out the truth table and the constructed incidence matrix of the original network by calling the subroutine OUTPUT. Finally the desired transduction



procedure is applied to the network by calling the subroutine(s) realizing that transduction. The transduced network is stored in INC\$MX, replacing the original network. Then MAIN prints the results of the transduction procedure, i.e., lists of removed and added connections, the new incidence matrix and the new truth table.

SUBNET: This subroutine may be entered at three different points by a call to either SUBNET, UNNECE, or PVALUE.

SUBNET generates detailed information on the configuration of the network stored in INC\$MX: (1) It calculates the level of each gate in the network. Level 1 is assigned to gates having no output connections thus all gates which have been removed from the network will be assigned level 1. (2) It lists all immediate successors and immediate predecessors for each gate. (3) It calculates the successor matrix which is stored in a two-dimensional array SUC\$MX. The value of SUC\$MX (A1,A2) indicates the existence or non-existence of a path from gate (or external variable) A1 to gate A2.

UNNECE disconnects certain types of obviously unnecessary connections in the network and updates the above information (discussed in (1), (2), and (3)). The connections removed from the given network are those existing in no paths from the external variables to the output gates.

PVALUE calculates the actual truth table for the entire network stored in INC\$MX.

OUTPUT: This subroutine may be entered at five different points by a call to either OUTPUT, PAGE, LINE, TRUTH, OR CKT.

OUTPUT assigns mnemonic names to external variables and gates for the purpose of achieving a readable print-out.

PAGE ejects one page on the printer.

LINE skips a specified number of lines on the print-out sheet. The number is specified by the argument in the call (e.g. "CALL LINE (5)" skips 5 lines).

TRUTH prints out the truth table of the network currently stored in INC\$MX.

CKT prints out the network itself.

MINI2: This subroutine is called by GTMERG in NETTRA-G3 for the purpose of calculating CSPF's for gates in a network. Subroutine MINI2 also has the ability to remove some redundant connections. It is described in some detail in [2].

## 5. INPUT DATA SETUP

In order to fully understand the description of the setup of the input data cards, certain preliminary explanations are necessary.

The purpose of network transductions is to reduce the cost of a network which realizes a certain function (or functions) or to alter the network in such a way as to allow another transduction to eventually accomplish such a reduction. This cost,  $C$ , is formally defined by the weighted sum of the number of gates,  $R$ , and the number of connections<sup>†</sup>,  $I$ , of a particular network, i.e.,

$$C = A \times R + B \times I$$

where weights  $A$  and  $B$  are arbitrary non-negative numbers.

Suppose the original network which is to be transformed produces  $m$  output functions of  $n$  variables. Let  $x_\ell$ ,  $\ell = 1, \dots, n$ , be the external variables and  $f_h$ ,  $h = 1, \dots, m$ , be the output functions. Before a transformation can be performed on a network by a program, a description of that network must be input to the program. In the case when all of the output functions are completely specified (i.e., no "don't cares"), specifying only the interconnection pattern of the network is sufficient. But if one or more of the output functions is not completely specified, then the user must also provide the truth table (truth tables for all output functions are condensed into a single table) of the problem. Providing the truth table to the program consists of two steps,

---

† A "connection" refers to either a connection from an external variable or an interconnection between two gates.

namely the specification of external variables, and the specification of output functions.

The method of specifying the output functions depends directly upon the method chosen to specify the external variables. External variables may be specified in either of two ways, (a) an implicit specification of external variables, or (b) an explicit specification of external variables.

(a) In the case of implicit specification of external variables, the user specifies the number  $n$  of external variables along with a parameter which indicates whether or not the uncomplemented variables are available. Reading the value  $n$  along with the parameter, the program internally generates the entries of external variables of an ordinary truth table, that is, a truth table which consists of  $2^n$  input vectors, as shown in Fig. 5.1. In this truth table, the input vectors are arranged according to the order such that an integer  $j$  expressed in a binary representation  $(x_1 \dots x_n)$  increases, where  $x_1$  is the most significant digit and  $x_n$  is the least significant digit. For example, the truth table for a function of three variables is shown in Fig. 5.2.

The implicit specification of external variables is used for logical design problems in which the output functions have relatively few don't-care terms.

The uncomplemented variables	$\left\{ \begin{array}{c} x_1 \\ \vdots \\ x_n \end{array} \right\}$	$x_1^0 \dots x_1^j \dots x_1^{2^n-1}$	
		$x_n^0 \dots x_n^j \dots x_n^{2^n-1}$	
The complemented variables	$\left\{ \begin{array}{c} \bar{x}_1 \\ \vdots \\ \bar{x}_n \end{array} \right\}$	$\bar{x}_1^0 \dots \bar{x}_1^j \dots \bar{x}_1^{2^n-1}$	
		$\bar{x}_n^0 \dots \bar{x}_n^j \dots \bar{x}_n^{2^n-1}$	
The output functions	$f_1$	$f_1^0 \dots f_1^j \dots f_1^{2^n-1}$	
	$f_2$		
	$\vdots$	$\vdots$	
	$f_m$	$f_m^0 \dots f_m^j \dots f_m^{2^n-1}$	

These entries exist only in the case of logical design problems where the complemented variables are available as external inputs.

Fig. 5.1 The truth table of output functions of  $n$  variables

$x_1$	0	0	0	0	1	1	1	1
$x_2$	0	0	1	1	0	0	1	1
$x_3$	0	1	0	1	0	1	0	1
$\bar{x}_1$	1	1	1	1	0	0	0	0
$\bar{x}_2$	1	1	0	0	1	1	0	0
$\bar{x}_3$	1	0	1	0	1	0	1	0
$f_1$	$f_1^0$	.	.	.	.	.	.	$f_1^7$

These entries exist only in the case of logical design problems where complemented variables are available as input variables.

Fig. 5.2 The truth table of a function of three variables.



(b) In the case of explicit specification of external variables, the user specifies the entries of external variables of the truth table using additional cards called < external-variable-card > s. The explicit specification of external variables is used in the case of logical design problems where output functions have many don't-care terms. Suppose that the output functions are defined for a subset of input vectors of the entire truth table of Fig. 5.1. Let  $\vec{x}^j$ ,  $j = j_1, j_2, \dots, j_\mu$  denote these input vectors. The user can 'condense' the truth table of Fig. 5.1 into another table shown in Fig. 5.3.

		only $\mu$ input vectors			
The uncomplemented variables	$x_1$	$x_1^{j_1}$	$x_1^{j_2}$	$\dots$	$x_1^{j_\mu}$
	$\vdots$	$\vdots$			
	$\vdots$	$\vdots$			
	$x_n$	$x_n^{j_1}$	$x_n^{j_2}$	$\dots$	$x_n^{j_\mu}$
The complemented variables	$\bar{x}_1$	$\bar{x}_1^{j_1}$	$\bar{x}_1^{j_2}$	$\dots$	$\bar{x}_1^{j_\mu}$
	$\vdots$	$\vdots$			
	$\vdots$	$\vdots$			
	$\bar{x}_n$	$\bar{x}_n^{j_1}$	$\bar{x}_n^{j_2}$	$\dots$	$\bar{x}_n^{j_\mu}$
	$f_1$	$f_1^{j_1}$	$f_1^{j_2}$	$\dots$	$f_1^{j_\mu}$
	$\vdots$	$\vdots$			
	$\vdots$	$\vdots$			
	$f_m$	$f_m^{j_1}$	$f_m^{j_2}$	$\dots$	$f_m^{j_\mu}$

These entries exist only in the case of logical design problems where the complemented variables are available as external inputs.

Fig. 5.3 A 'condensed' truth table having only the input vectors  $\vec{x}^j$ ,  $j = j_1, \dots, j_\mu$ , for which the output functions are defined.

Using < external-variable-card > s, the user can set up internally the table of Fig. 5.3 in place of Fig. 5.1.

### 5.1 Input Data Card Format

For each separate problem, a set of input data cards must be submitted which consists of the following<sup>†</sup>:

- |       |                                   |                                |
|-------|-----------------------------------|--------------------------------|
| (i)   | < heading-card >                  |                                |
| (ii)  | < problem-parameter-card >        |                                |
| (iii) | < external-variable-card > s      | } omitted for<br>certain cases |
| (iv)  | < output-function-card > s        |                                |
| (v)   | < connection-description-card > s |                                |

Both (i) and (ii) will always consist of only a single card, but (iii), (iv), and (v) may each consist of several cards. Furthermore, types (iii) and (iv) are omitted if all output functions are completely specified, and (iii) need only be prepared in the case of the explicit specification of external variables for the truth table. Following is a description of the formats for each type of input card, (i), (ii), (iii), (iv) and (v):

#### (i) < Heading-card >

This is the first card of the input deck for a problem. This card may contain any alphanumeric information, in columns 1~80, which may be used for the identification of the problem, but none of the information on this card will be used in the actual computation. This information will be printed on the first page of the output.

---

† The current implementations of the NETTRA programs accept only heading, problem-parameter, and connection-description cards. Eventually it is hoped that these programs will be modified to accept all of the options described in this section.



## (ii) &lt; Problem-parameter-card &gt;

This card specifies the nature of the problem the user wants to solve. There are 7 fields in which to specify the parameters with characters and numerals. These fields are as follows:

Cols. 1~4: An integer,  $N$ , which is right-justified.

This number,  $N$ , represents the number of external variables,  $n$ , of the output functions. Be sure to punch  $n$  (rather than  $2n$ ) for  $N$  in the case of both complemented and uncomplemented variables available.

Cols. 5~8: An integer,  $M$ , which is right-justified.

This number,  $M$ , is the number of output functions,  $m$ , to be realized simultaneously. Therefore, of course,  $M$  will also be the number of output gates in the network.

Cols. 9~12: An integer,  $R$ , which is right-justified.

This number,  $R$ , specifies the number of gates which are included in the network. For various reasons, the user may wish to input networks in which one or more of the gates are "isolated" (i.e., are not connected to any other gates). This is permissible as long as these "isolated" gates are also included in the total number of gates,  $R$ .

Cols. 13~16: An integer,  $A$ , which is right-justified.

The number  $A$  is the value of the non-negative weight for the number of gates in the cost function. (See Table 5.1.1, 'Typical combinations of values  $A$  and  $B$  for different network reduction problems'.)

Cols. 17~20: An integer, B , which is right-justified.

The number B is the value of the non-negative weight for the number of connections in the cost function. (See Table 5.1.1.)

Col. 24: A blank 'b'<sup>†</sup>, or one of the characters, 'C', 'X', 'Y', 'U' or 'V'.

The 'b' or 'C' parameter represents an implicit specification of both the external variables and an implicit specification of the output functions (in this case, the output functions will be calculated from the connection pattern of the network). The 'X' or 'Y' parameter indicates an implicit specification of external variables only. The 'U' or 'V' parameter indicates an explicit specification of external variables. (See summary of these symbols in Table 5.1.2)

The 'b' or 'X' parameter specifies that only uncomplemented external variables are available for the network. The 'C' or 'Y' parameter specifies that both uncomplemented and complemented variables are available for the network. If the user specifies the 'b', 'X', 'C', or 'Y' parameter, the program sets up the truth table by generating a set of  $2^n$  input vectors  $(x_1^j, \dots, x_n^j)$ , for  $j = 0, \dots, 2^n - 1$ , in the case of a 'b' or 'X' parameter, or  $(x_1^j, \dots, x_n^j, \bar{x}_1^j, \dots, \bar{x}_n^j)$  for  $j = 0, \dots, 2^n - 1$ , in the case of a 'C' or 'Y' parameter.

The 'b' or 'C' parameters should be used for problems in which the output functions contain no don't-care terms. For such problems, the preparation of the < external-variable-card > s and the < output-function-card > s can be dispensed with since the program can calculate completely all output functions using only a description of the

---

<sup>†</sup> A 'b' stands for a blank (i.e., no character punched).

Network Reduction Problem	Values of A and B
reducing the number of gates only.	$A = 1$ and $B = 0$
reducing the number of gates primarily, then reducing the number of connections secondarily. <sup>†</sup>	$A = 100$ and $B = 1$
reducing the number of connections only.	$A = 0$ and $B = 1$
reducing the number of connections primarily, then reducing the number of gates secondarily.	$A = 1$ and $B = 100$
reducing the sum of the number of gates and the number of connections.	$A = B = 1$

Table 5.1.1 Typical combinations of values A and B for different network reduction problems.

<sup>†</sup> Most of the programs in the NETTRA system are oriented toward this reduction problem, so the user will probably find this combination of A and B the most useful.

uncomplemented variables only available	both complemented and uncomplemented variables available	
'b'	'c'	} implicit specification of external variables and output functions
'x'	'y'	
'u'	'v'	} explicit specification of external variables

Table 5.1.2 Possible symbols for column 24 of < problem-parameter-card >.

connection pattern of the network (provided by the <connection-description-card>s).

Similarly, the 'X' or 'Y' parameter implies the use of a complete truth table (i.e.,  $2^n$  input vectors for  $n$  external variables) inside the program. Since from this information the program can easily generate the truth table entries for the external variables, as just mentioned, the < external-variable-card > s are unnecessary. The  $m$  < output-function-card > s, however, must still be prepared.

The 'U' parameter specifies that only uncomplemented external variables are available for the network. The 'V' parameter specifies that both uncomplemented and complemented variables are available for the network. In either case, the 'U' or the 'V' parameter, the user must prepare  $n$  < external-variable-card > s and  $m$  < output-function-card > s. The program sets up the truth table by reading these < external-variable-card > s and < output-function-card > s.

Cols. 25~28: An integer, NEPMAX, which is right-justified.

This parameter is omitted for all NETTRA programs except those involving "error-compensation" routines. In the cases where NEPMAX is required, a further discussion of this parameter can be found elsewhere in the manual. The abbreviation NEPMAX is a mnemonic for "maximum number of error positions", and the default is  $NEPMAX = 2^{(n-1)}$ , where  $n$  is the number of external variables.

(iii) < External-variable-card > s

In combination with the 'U' or 'V' parameter in column 24 of the < problem-parameter-card >, the  $n$  < external-variable-card > s specify the entries of external variables of the truth table of



Fig. 5.3. Each card contains the binary representation of external variable  $x_\ell$ , i.e.,  $(x_\ell^{j1}, x_\ell^{j2}, \dots, x_\ell^{j\mu})$ , starting from column 1 of the card. The maximum number of bits in a binary representation is limited to 32. (This means the maximum number of input vectors is 32.) If the actual number of bits is less than 32, then a termination symbol '/' (slash) is put on the right of the right-most bit of the binary representation on the first < external-variable-card >. The remaining columns after the termination symbol '/' in the first card, as well as the same columns in the following cards, may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

In the case of the 'V' parameter, the program generates the binary representations corresponding to complemented variables by taking negations of the entries of the < external-variable-card > s. Therefore the user must not provide < external-variable-card > s representing the complemented variables,  $\bar{x}_\ell$ .

If one of the parameters 'b', 'C', 'X', or 'Y' appears in column 24 of the < problem-parameter-card >, the user does not prepare < external-variable-card > s.

(iv) < Output-function-card > s

The  $m$  < output-function-card > s specify the set of  $m$  output functions to be realized simultaneously. Each card contains the binary representation of one output function  $f_h$ , starting from column 1 of the card. A symbol '\*' is used to denote don't-care terms, if any. The maximum number of bits in a binary representation is limited to 32.

The actual number of bits must be  $2^n$  in the case of an implicit specification of external variables, or must be the same as defined by the < external-variable-card > s in the case of an explicit specification of external variables. The remaining columns, up to column 72 (inclusive), may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

If either the 'b' or 'C' parameter appears in column 24 of the < problem-parameter-card >, the < output-function-card > s must be omitted.

(v). < Connection-description-card > s

In the present version of the program, 9 cards (some of which may be just blank cards) are required.<sup>†</sup> Each of these cards is divided into 16 fields of 5 columns each (i.e., columns 1~5, 6~10, 11~15, ..., 71~75, 76~80). Beginning with the first field of the first card, continuing through the succeeding fields of that card and through the fields of as many additional cards as necessary (up to a maximum of 9, total), the expressions (explained in the next paragraph)  $C_1, C_2, C_3, \dots$ , are punched right-justified in their respective fields.

Each gate of the network is labeled uniquely by assigning it one of the integers 1, 2, ..., R, such that the output gates receive

---

† For many uses, the user will probably find that these 9 cards far exceed his needs, and may thus be inconvenient. In such a case, the number of required cards may be easily adjusted by making the obvious changes in two statements (A READ statement and a DO statement) following the comment card "C\*\*\*\* READ IN NETWORK INFORMATION AND SET UP INC\$MX \*\*\*\*\*" in subroutine MAIN.



the labels 1, 2, ..., m. The names  $X_1, X_2, \dots, X_n$  are assigned to the external variables  $x_1, x_2, \dots, x_n$  (and the names  $Y_1, Y_2, \dots, Y_n$  to the complemented external variables  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ , if appropriate).<sup>†</sup> Now, for each connection of the network (i.e., including both the connections from external variables to gates and connections from gates to other gates), a 4 character expression,  $C_i$ , is formed, to represent that connection as follows: to represent a connection from gate GI to gate GJ, the numeric label GI is inserted into the first two character positions of  $C_i$  and the numeric label GJ is inserted into the second two positions (e.g., the  $C_i$  for a connection from gate 9 to gate 5 would be "0905"); to represent a connection from external variable XI to gate GJ, the alphanumeric label XI is inserted into the first two character positions of  $C_i$  and the numeric label GJ into the second two positions (e.g., the  $C_i$  for a connection from external variable  $x_3$  to gate 10 would be "X310").

Every connection of the network must be represented by a  $C_i$ , although there are no restrictions on the order in which the connections (i.e.,  $C_i$ 's) are punched onto the input cards.

---

<sup>†</sup> At the time of writing, the programs have not yet been changed to recognize this new labeling system. The old labels are simply:

1, 2, ..., n, for external variables  $x_1, x_2, \dots, x_n$  (and  $n+1, n+2, \dots, 2n$  for the complemented variables  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ , if they are permitted in the problem);  $n+1, n+2, \dots, n+m$  for the m output gates of the network ( $2n+1, 2n+2, \dots, 2n+m$  if complemented variables are included); and finally  $n+m+1, n+m+2, \dots, n+R$  ( $2n+m+1, 2n+m+2, \dots, 2n+R$ ) for the non-output gates of the network.

These five groups of cards, (i), (ii), (iii), (iv) and (v) in the correct order constitute the necessary description for a single problem. In order to solve several problems during the same computer run, the descriptions for the desired problems are just arranged serially. See Fig. 5.1.1 for an example of the sequencing of all cards for the execution of a NETTRA program using typical JCL statements for the IBM 360/75.

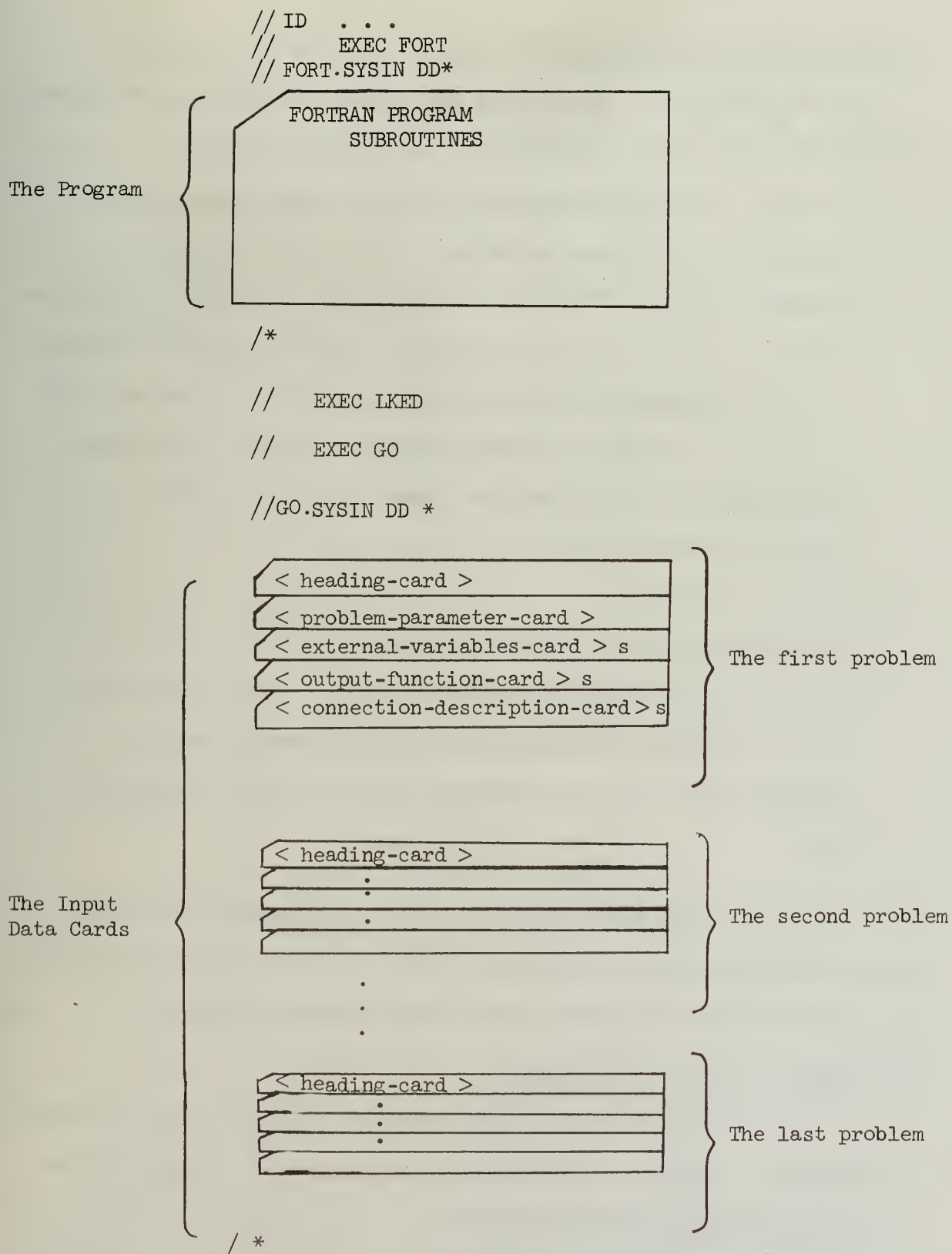


Fig. 5.1.1 Input card sequence for the execution of a typical NETTRA program.

## 5.2 Restrictions on Problem Size

In order to fit the programs into a finite amount of space, some restrictions on the size of an acceptable problem are required:

1. The number  $t$  of input vectors in the truth table is 32 or less.
2. The number  $n$  of external variables.

Because of  $t \leq 32$ ,  $n$  is 5 or less in the case of completely specified functions. In the case of incompletely specified functions, however, any  $n \leq 20$  is acceptable if only uncomplemented variables are available, or  $n \leq 10$  if both uncomplemented and complemented variables are available, provided that the truth table is defined by the < external-variable-card > s.

3. The number  $R$  of gates.

The number of gates,  $R$ , may not exceed  $40-n$  in the case of only uncomplemented variables available (a 'b', 'X', or 'U' parameter).

In the case of both uncomplemented and complemented variables available (a 'C', 'Y' or 'V' parameter), the limit is lowered to  $40-2n$ .

All of these limitations are essentially imposed by the array sizes in the programs as presently written. To loosen the restrictions is mainly a task of increasing array dimensions appropriately.

## 5.3 Examples of Input Data Setup

The following examples will illustrate, for the general program in the NETTRA system, various possible input data card setups complying with the directions given in Section 5.1.

Example 1: A two output network of four variables shown in Fig. 5.3.1. Assume the two output functions are  $f_1 = CCE\overline{F}$ <sup>†</sup> and  $f_2 = 3BBB$  and only uncomplemented variables are available. Furthermore, assume it is desired to reduce the number of gates primarily and the number of connections secondarily (see Table 5.1.1).<sup>††</sup>

From this description, the < problem-parameter-card > must contain the following values:

Cols. 1~4	4, the number of external variables
Cols. 5~8	2, the number of output functions
Cols. 9~12	8, the number of gates in the original network
Cols. 13~16	100, the value of A
Cols. 17~20	1, the value of B
Cols. 24	'b', uncomplemented variables only available and implicit specification of both the external variables and the output functions
Cols. 25~28	'b', since the NEPMAX parameter is unrelated to the program to be used

Fig. 5.3.2 shows the setup of data cards used to specify the network in Fig. 5.3.1 as input for the program. Notice that in forming the  $C_i$ , the four uncomplemented variables are represented by the labels  $X_1, X_2, X_3, X_4$ ; the two output gates by the numbers 1, 2; and the remaining gates, by the numbers 3, 4, 5, 6, 7, 8. This manner of labeling is

---

† For convenience, the output functions are expressed in hexadecimal notation. When the numbers in this notation are expanded into binary, they represent the output vectors as they appear (i.e., in the same left-to-right order) in the complete truth table described earlier and pictured in Fig. 5.1.

†† This assumption is implicit in most of the transduction procedures and their implementations which comprise the NETTRA system.

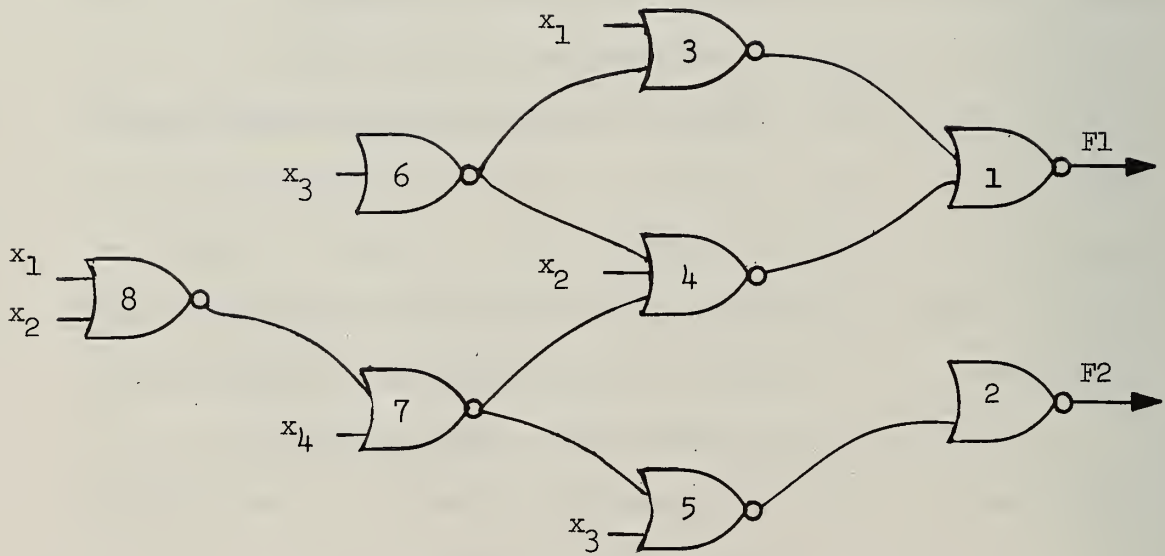


Fig. 5.3.1 Network to be transformed in Examples 1 and 2.







strictly required by the instructions for preparing the < connection-description-card > s (see Section 5.1).

The heading card in Fig. 5.3.2 will simply be read by the program and printed character for character onto the output page as an identification of the particular problem. Below that, the number of variables, number of functions, and the cost coefficients, A and B, will be printed (all with appropriate labels). Also, immediately following will be a statement of what types of external variables are permitted (i.e., either just uncomplemented variables or both complemented and uncomplemented) along with their generic names:

X - for uncomplemented variables	}	if external variables were implicitly specified
Y - for complemented variables		
<u>or</u>		
U - for uncomplemented variables	}	if external variables were explicitly specified
V - for complemented variables		

For example, if both X and Y appear as generic names (as would occur in the case of an implicit specification of external variables with both complemented and uncomplemented variables available) then the external variable names which appear on subsequent output pages will be X1, X2, ..., Xn and Y1, Y2, ..., Yn. Or, if both U and V appear as generic names (as would occur in the case of an explicit specification of external variables with both complemented and uncomplemented variables available) the external variable names which appear in the output will be U1, U2, ..., Un (for the uncomplemented variables) and V1, V2, ..., Vn (for the complemented variables). It should be noted, however, that the letters U and V, as used as replacements for X and Y (respectively) in the

naming of external variables (e.g. U1, V1 instead of X1, Y1), appear strictly on the output pages of the program - they are not used internally in the program and they must not appear in the variable names punched on the < connection-description-card > s by the user. They are intended only as an aid to the user so that, at a glance at the transformed network in the output, he can easily distinguish whether the external variables were implicitly or explicitly specified for that particular problem.

Following the statement of whether only uncomplemented or both complemented and uncomplemented external variables are employed, the user will find next on the output page the cost of the original network which was input to the program. This is the cost which was defined in the beginning of Section 5.

The cost will be followed by a truth table (generally in the same form as Fig. 5.1) showing the outputs (0 or 1) of all of the gates in the network for every external variable input combination (i.e., combinations of 0's and 1's) of interest.

Finally, below the truth table will be printed a description of the network submitted as input. This is for documentation purposes, and it is also much more readable than the network description which appeared on the < connection-description-card > s. In this description, each gate is listed along with the names of the gates and external variables which feed it. Also, to assist the user in sketching the network from its description, the level of each gate in the network is included (gates which do not feed other gates are assigned to level 1, all other gates are assigned level numbers such that each gate is in a level one

higher than the highest level gate directly fed by it).

All of the information just described will be printed before the execution of the transduction actually begins. This will be followed, beginning at the top of a new output page, by the network(s) actually obtained as a result of the computation. First the complete truth table of the transformed network will be printed, followed by a network connection description of the form just described above. Finally, the cost of the new network will be calculated and printed.

In this example, it was assumed that there were no "don't-cares" in the output functions implicitly specified by the input, thus no < external-variable-card > s or < output-function-card > s were included. In the next example, however, < output-function-card > s will be required in order to specify some of the components of the output functions as "don't-cares".

Example 2: The two output network of four variables shown in Fig. 5.3.1. This is the same network used in Example 1, but this time the output functions are not assumed to be completely specified. Let  $f_1 = '11001**01*10*111'$  and  $f_2 = '0**110111*111011'$  be the required functions. Also, suppose that both complemented and uncomplemented variables are desired to be available during the transduction. Again the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this problem, the following values must appear on the < problem-parameter-card >:

Cols. 1~4      4, the number of external variables

Cols. 5~8      2, the number of output functions

Cols.	9~12	8, the number of gates in the original network
Cols.	13~16	100, the value of A
Cols.	17~20	1, the value of B
Col.	24	Y, indicative of an implicit specification of external variables and the availability of both complemented and uncomplemented variables

Fig. 5.3.3 shows the setup of the data cards corresponding to this problem. Notice the differences and similarities to the data cards shown in Fig. 5.3.2. The < problem-parameter-card > differs only in column 24. The < external-variable-card > s are missing in both Fig. 5.3.2 and Fig. 5.3.3 since the external variables are implicitly specified for both problems. The < output-function-card > s, however, appear in Fig. 5.3.3 but not in 5.3.2 since they are necessary to specify "don't-care" components which do not occur in the completely specified output functions of Example 1. In both cases, though, the < connection-description-card > s are identical since the original networks are identical.

By allowing "don't-care" terms in the output functions, and by allowing the use of both complemented and uncomplemented variables<sup>†</sup> (even though the original network employed only uncomplemented variables), the restrictions during the transduction process are loosened (compared to what they were for Example 1), perhaps permitting a network of

---

<sup>†</sup> In the case of NETTRA-PG1, -P1, and -P2, it is useless to specify Y rather than X in column 24 for this example. Since the original network uses only uncomplemented variables, to these programs which perform "pruning" procedures (i.e., procedures which are incapable of adding new connections) the availability of complemented variable is not meaningful.

Column No.: 00000000011  
12345678901...

...78  
...90

8 blank connection- description-cards	.
	.
	.
1st connection- description-card	__301__401__502__X103__603__604__X204__704__705__X305__X306__807__X407__X108__X208__
2nd output- function-card	0**110111*111011-----REQUIRED OUTPUT FOR GATE 2
1st output function-card	11001**01*10*111-----REQUIRED OUTPUT FOR GATE 1
problem- parameter-card	4 2 8 100 1 Y -----
heading-card	2 OUTPUT, 4 VARIABLE NETWORK, F1=11001**01*10*111, F2=0**110111*111011

Fig. 5.3.3 Possible setup of data cards to specify the problem given in Example 2.



less cost to be obtained.

Notice that the first < output-function-card > corresponds to the output of gate 1 and the second < output-function-card > corresponds to the output of gate 2. This must hold true for every problem in which < output-function-card > s are included; the gates labeled 1, 2, ..., m must correspond to the output functions specified on < output-function-card > s 1, 2, ..., m, respectively.

Of course, the printed output of the program will be in the same format described in Example 1.

Example 3: The three output network of six variables shown in Fig. 5.3.4. The outputs are again assumed to be incompletely specified. In fact, only the following 11 input combinations are specified out of a possible  $64 (=2^6)$ :

$x_1$	0 0 0 0 0 0 0 0 0 0 0 1
$x_2$	0 0 0 0 0 0 0 0 1 1 1 0
$x_3$	0 0 0 0 0 0 0 0 0 0 1 1
$x_4$	0 0 0 0 1 1 1 0 0 0 0 1
$x_5$	0 0 1 1 0 0 1 0 1 1 0
$x_6$	0 1 0 1 0 1 0 1 1 0 0
<hr/>	
$F_1$	0 0 1 1 0 0 * 0 0 0 0
$F_2$	1 1 * 1 1 1 0 1 1 0 *
$F_3$	1 1 0 0 0 0 0 1 0 0 0

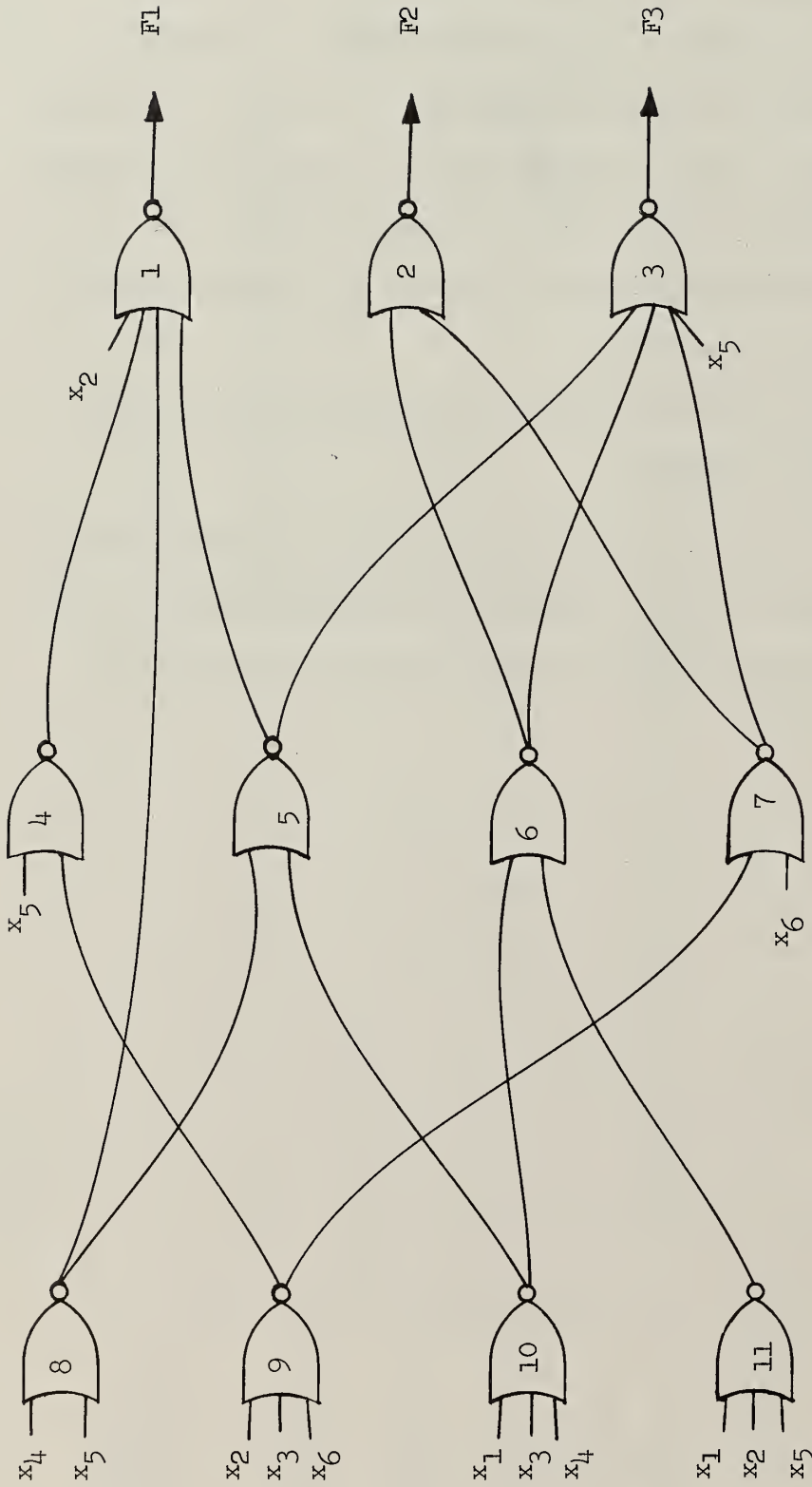


Fig. 5.3.4 Network to be transformed in Example 3.



Additionally, only uncomplemented variables are assumed to be available, and the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this example, the following parameters appear on the  
< problem-parameter-card >:

Cols.	1~4	6, the number of external variables
Cols.	5~8	3, the number of output functions
Cols.	9~12	11, the number of gates in the original network
Cols.	13~16	100, the value of A
Cols.	17~20	1, the value of B
Col.	24	U, indicative of an explicit specification of external variables and the availability of only uncomplemented variables

Fig. 5.3.5 shows a possible setup of the data cards corresponding to this example. Notice that in this example, the <external-variable-card> s are included, whereas in the two previous examples they were omitted. Although this problem is not too realistic (none of the 3 functions is actually a 6-variable function), it demonstrates the input data preparation to be used in cases where many external variables are present and a high percentage of "don't care" terms exist.

Again, the printed output from the program will follow the same format described in Example 1.



## REFERENCES

- [1] H.C. Lai and Y. Kambayashi, "NOR Network Transduction by Generalized Gate Merging and Substitution (Principles of NOR Network Transduction Programs NETTRA-G3 and NETTRA-G4)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.
- [2] H. C. Lai and J. N. Culliney, "Program Manual: NOR Network Pruning Procedures Using Permissible Functions (Reference Manual of NOR Network Transduction Programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.
- [3] Y. Kambayashi and S. Muroga, "Network Transduction Based on Permissible Functions (General Principles of NOR Network Transduction NETTRA Programs)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.
- [4] J. N. Culliney, H. C. Lai, and Y. Kambayashi, "Pruning Procedures for NOR Networks Using Permissible Functions (Principles of NOR Network Transduction Programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

## APPENDIX

### Program Listings

Following are the FORTRAN program listings for the two programs, NETTRA-G3 and NETTRA-G<sup>4</sup>, respectively, discussed in this manual. Many of the variables and arrays used are defined in the program themselves.

```

*****
      PPPP      RRRR      OOO      GGG      RRRR      A      M      M
      P  P      R  R      O  O      G  G      R  R      A  A      MM  MM
      P  P      R  R      O  O      G  G      R  R      A  A      M M M M
      PPPP      RRRR      O  O      G  GG      RRRR      AAAAA      M  M  M
      P          R  R      O  O      G  G      R  R      A  A      M  M
      P          R  R      OOO      GGG      R  R      A  A      M  M
*****

```

```

C N  N  EEEEE  TTTT  TTTT  RRRR      A      GGG      33333
C NN  N  E      T      T      R  R      A  A      G  G      3
C N  N  N  E      T      T      R  R      A  A      G      3
C N  NN  EEE      T      T      RRRR      AAAAA  XXXXX  G  GG      3
C N  N  E      T      T      R  R      A  A      G  G      3  3
C N  N  EEEEE      T      T      R  R      A  A      GGG      333
*****

```

```

IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                                G3 00010
EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB G3 00020
                                                                 G3 00030
NOTE: ALL COMMON VARIABLES MIGHT NOT BE USED IN THIS PROGRAM.        G3 00040
                                                                 G3 00050
COMMON VARIABLES:                                                       G3 00060
    $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY      G3 00070
           IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.             G3 00080
    $LTH: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY      G3 00090
           IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.      G3 00100
    $NOE: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY      G3 00110
           IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS   G3 00120
           ROW IS USED.                                                  G3 00130
    $PW: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY      G3 00140
           IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.                  G3 00150
    A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.             G3 00160
    B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.       G3 00170
    CGST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.                 G3 00180
    ESSIS: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO G3 00190
           (POSITIONS IN ESSIS CORRES. TO GATES NOT FEEDING GCO ARE    G3 00200
           IGNORED).                                                    G3 00210
    F$UB1: POINTS TO LAST ELEMENT IN F$1.                               G3 00220
    F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR        G3 00230
          COVERING) IN A CONNECTIBLE FUNCTION.                         G3 00240
    GI: LABEL OF A PARTICULAR GATE.                                     G3 00250
    GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.     G3 00260
    G$SMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.          G3 00270
    HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-   G3 00280
           WORK LEVEL J.                                                G3 00290
    IDX0: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER            G3 00300
           CONSIDERATION.                                               G3 00310
    IDXOE: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER     G3 00320
           CONSIDERATION.                                               G3 00330
    IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER            G3 00340
           CONSIDERATION.                                               G3 00350
    IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER    G3 00360

```



	CONSIDERATION.	G3 003
IFLAG:	SAME AS EYEFLG IN SUBROUTINE PROCII.	G3 003
INC\$MX:	INC\$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE (OR EX. VAR.) GI TO GATE GJ. INC\$MX(GI,GJ)=0 IF NOT.	G3 003
INPTCV:	LISTS FOR EACH CORRESPONDING ENTRY OF F\$1, HOW MANY INPUTS HAVE A '1' IN THE POSITION INDICATED BY F\$1.	G3 004
IPATH:	IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATE TO AN OUTPUT GATE. OTHERWISE IPATH(GI) = 0.	G3 004
IPRED:	IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. A LIST OF GATES AND EX. VAR. FEEDING GJ.	G3 004
ISUCC:	ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.	G3 004
JFLAG:	SAME AS JAYFLG IN SUBROUTINE PROCII.	G3 004
KEYA:	A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN PERFORMED.	G3 004
KEYB:	A FLAG INDICATING IF ANY PRIMARY O-ERROR-COORDINATES HAS BEEN COMPENSATED.	G3 005
KFLAG:	SAME AS KEIFLG IN PROCII.	G3 005
LEVLM:	NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO ASSIGNED LEVELS JUST LIKE GATES).	G3 005
LGLIST:	LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF NETWORK.	G3 005
LIP:	NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.	G3 005
LIPRED:	LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.	G3 005
LISTC:	ORDERED LIST OF CONNECTIBLE INPUTS TO GCO. ORDERED BY DECREASING NO. OF O'S IN GCO COVERED.	G3 006
LISTL:	ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED. ORDERED BY DECREASING NO. OF ESSENTIAL 1'S.	G3 006
LISUCC:	LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR EX. VAR.) GI.	G3 006
LMTS2:	UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.	G3 006
LPOTAB:	FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB CONCERNING GI.	G3 006
M:	NUMBER OF NETWORK OUTPUT GATES.	G3 007
N:	NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.	G3 007
NEPMAX:	FOR ERROR COMPENSATION PROGRAMS. IF MORE THAN NEPMAX ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED, PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S REMOVAL. VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL TO ONE HALF OF N2 BY DEFAULT.	G3 007
NM:	SUM OF N PLUS M	G3 007
NM1:	SUM OF NM PLUS 1.	G3 007
NN2:	PRODUCT OF N AND N2.	G3 007
NOS:	NUMBER OF ELEMENTS IN SET S.	G3 008
NOS1:	NUMBER OF ELEMENTS IN SET S1.	G3 008
NOSISV:	NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE RPLCF.	G3 008
NOS2:	NUMBER OF ELEMENTS IN SET S2.	G3 008
NOT1:	NUMBER OF ELEMENTS IN SET T1.	G3 008
NOTISV:	NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE RPLCF.	G3 008
NOO:	NUMBER OF ACTIVE ELEMENTS IN ARRAY IDXO.	G3 008
NOOE:	NUMBER OF ACTIVE ELEMENTS IN ARRAY IDXOE.	G3 008
NO1:	NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.	G3 009
NO1E:	NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.	G3 009
NR:	SUM OF N PLUS R.	G3 009
NRN2:	PRODUCT OF NR AND N2.	G3 009
NRPLC:	NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*) FOR I=1,2.	G3 009
N1:	SUM OF N PLUS 1.	G3 009
N2:	NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED	G3 009



(USUALLY 2 TO THE POWER N).

ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO. G3 00980  
ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY. G3 00990  
G3 01000  
P\$: P\$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND G3 01010  
EVERY GATE (FOR EVERY INPUT COMBINATION): P\$(1,1),..., G3 01020  
P\$(1,N2) FOR FIRST EX VAR; P\$(1,N2+1),...,P\$(1,2\*N2) FOR G3 01030  
SECOND EX VAR; ... ; P\$(1,N\*N2+1),..., P\$(1,N\*N2+N2) FOR G3 01040  
FIRST GATE; ETC. P\$(2,-) IS USED AS WORK SPACE FOR G3 01050  
CALCULATIONS ASSOCIATED WITH P\$(1,-). G3 01060  
PCO: FOR ERROR COMPENSATION PROCEDURES. PCO IS THE GATE G3 01070  
REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED G3 01080  
NETWORK. G3 01090  
POINTA: NOT USED. G3 01100  
POINTC: POINTS TO LAST ELEMENT IN LISTC. G3 01110  
POINTL: POINTS TO LAST ELEMENT IN LISTL. G3 01120  
POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBST1). G3 01130  
POTAB: POSSIBLE OUTPUT TABLE. HOLDS INFORMATION ABOUT ALL G3 01140  
COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY G3 01150  
USEFUL) FUNCTIONS. G3 01160  
PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF G3 01170  
ROWS OF POTAB CONCERNING GI. G3 01180  
R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO G3 01190  
NOTE SOME OF R GATES MAY BE ISOLATED). G3 01200  
RPLC: RPLC(1,\*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE G3 01210  
ERROR-COORDINATES OF WEIGHT 2 OR ABOVE. G3 01220  
RPLC(2,\*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE G3 01230  
AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1. G3 01240  
RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM). G3 01250  
RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM). G3 01260  
S: NO. OF CONNECTIONS ADDED TO A NETWORK. POINTS TO LAST G3 01270  
ENTRY IN RSCONN. G3 01280  
SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATION G3 01290  
WHICH ARE TO BE REPLACED IF POSSIBLE. G3 01300  
SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE G3 01310  
REPLACED BY ELEMENTS IN SET S2. G3 01320  
SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR G3 01330  
REPLACING ELEMENTS IN SET S. G3 01340  
SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS IN G3 01350  
SET S1. G3 01360  
STS: STARTING ELEMENT OF SET S. G3 01370  
SUC\$MX: SUC\$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI. G3 01380  
SUC\$MX(GI,GJ)=0 IF NOT. G3 01390  
SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION. G3 01400  
SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2. G3 01410  
T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK. POINTS TO G3 01420  
LAST ENTRY IN RTCONN. G3 01430  
TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME. G3 01440  
UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES. G3 01450  
VF\$UB1: POINTS TO LAST ELEMENT IN VF\$1. G3 01460  
VF\$1: SIMILAR TO F\$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONS G3 01470  
(OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING G3 01480  
ORIGINALLY CONNECTED INPUTS TO GCO. G3 01490  
G3 01500  
G3 01510  
G3 01520  
G3 01530

COMMON NEPMAX  
COMMON N , M , A , B G3 01540  
1 , R , N2 , N1 , NR G3 01550  
2 , NM , KFLAG , JFLAG , COST G3 01560  
3 , LEVM , NRN2 , NM1 , NN2 G3 01570  
COMMON ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40) G3 01580

```

1      , INC$MX(40,40), SUC$MX(40,40), P$(2,1280) , UNAME(40) G3 01590
2      , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME G3 01600
COMMON T , RTCONN(100) , S , RSCONN(100) G3 01610
COMMON IFLAG , POINTA , ESSIS(40) , F$1(32) G3 01620
1      , F$UB1 , INPTCV(32) , LISTC(40) , POINTC G3 01630
2      , LISTL(40) , POINTL , ORIGIN(40) , IPATH(40) G3 01640
3      , POINTR , VF$1(32) , VF$UB1 , GSMALL(40,32) G3 01650
COMMON POTAB(200,42), PPOTAB(40) , LPOTAB(40) , NRPLC(2) G3 01660
1      , RPLC(2,40) , IDX0(32) , IDX0E(32) , IDX1(32) G3 01670
2      , IDX1E(32) , SUMP(32) , SETT1(32) , NOT1 G3 01680
3      , SETS1(40) , NOS1 , SETS(40) , NOS G3 01690
4      , STS , SUMS2(32) , SETS2(200) , NOS2 G3 01700
5      , LIP , NOOE , KEYA , KEYB G3 01710
6      , NCO , NO1 , NO1E , $GT G3 01720
7      , $LTH , $PW , $NOE , GI G3 01730
COMMON NOTISV , NOSISV , LMTS2 G3 01740
DIMENSION CNTLIS(144),UGATE(40),UHEAD(20) G3 01750
DATA KOUNT5 /0/, UBLANK/' ' G3 01760
990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX G3 01770
NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS G3 01780
1000 FORMAT(20A4/5I4,A4,I4) G3 01790
KEYXC=0 G3 01800
IF(UC.NE.UBLANK) KEYXC=1 G3 01810
CALL PAGE G3 01820
CALL LINE(10) G3 01830
KCOUNT5=KCOUNT5+1 G3 01840
PRINT 2, KCOUNT5 G3 01850
2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 ) G3 01860
CALL LINE(4) G3 01870
PRINT 1005, UHEAD G3 01880
1005 FORMAT(25X,20A4) G3 01890
CALL LINE(4) G3 01900
PRINT 10, N,M,A,B G3 01910
10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 // G3 01920
1 30X,'NUMBER OF FUNCTIONS =',I4 // G3 01930
2 30X,'COST COEFFICIENT A =',I4// G3 01940
3 47X, 'B =',I4) G3 01950
CALL LINE(1) G3 01960
IF(KEYXC.NE.0) GO TO 25 G3 01970
PRINT 21 G3 01980
21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES X ---') G3 01990
GO TO 30 G3 02000
25 CONTINUE G3 02010
PRINT 28 G3 02020
28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLES G3 02030
1 X, Y ---') G3 02040
30 CONTINUE G3 02050
CALL LINE(5) G3 02060
***** SET UP EXTERNAL VARIABLES ***** G3 02070
N2=2*N G3 02080
IF(NEPMAX.EQ.0) NEPMAX = N2/2 G3 02090
H=N*N2 G3 02100
J=N2 G3 02110
L= 1 G3 02120
I=0 G3 02130
DO 1011 II=1,N G3 02140
J=J/2 G3 02150
L=L*2 G3 02160
SN= 1 G3 02170
DO 1010 LL=1,L G3 02180
SN=-SN G3 02190

```

V=(1+SN)/2	G3 02200
DO 1009 JJ=1,J	G3 02210
I=I+1	G3 02220
P\$(1,I)=V	G3 02230
IF(KEYXC.NE.0)P\$(1,I+H)=1-V	G3 02240
009 CONTINUE	G3 02250
010 CONTINUE	G3 02260
011 CONTINUE	G3 02270
IF(KEYXC.NE.0) N=N+N	G3 02280
N1=N+1	G3 02290
NM=N+M	G3 02300
NM1=N+1	G3 02310
NN2=N*N2+1	G3 02320
NR=N+R	G3 02330
NRN2=NR*N2	G3 02340
CALL OUTPUT(INC\$MX,KEYXC)	G3 02350
**** READ IN NETWORK INFORMATION AND SET UP INC\$MX ****	G3 02360
READ 1001, CNTLIS	G3 02370
001 FORMAT(16I5)	G3 02380
DO 1115 GI=1,NR	G3 02390
DO 1115 GJ=1,NR	G3 02400
115 INC\$MX(GI,GJ)=0	G3 02410
DO 1120 I=1,144	G3 02420
ITEM=CNTLIS(I)	G3 02430
IF(ITEM.EQ.0) GO TO 1119	G3 02440
GI=ITEM/100	G3 02450
GJ=ITEM-100*GI	G3 02460
INC\$MX(GI,GJ)=1	G3 02470
GO TO 1120	G3 02480
119 COST=A*R+B*(I-1)	G3 02490
GO TO 1130	G3 02500
120 CONTINUE	G3 02510
130 CONTINUE	G3 02520
CALL SURNET	G3 02530
CALL PVALUE	G3 02540
CALL LINE(4)	G3 02550
PRINT 1140, COST	G3 02560
140 FORMAT(20X,' ORIGINAL NETWORK COST=', 15)	G3 02570
CALL LINE(4)	G3 02580
CALL TRUTH(P\$,1)	G3 02590
CALL LINE(4)	G3 02600
CALL CKT(INC\$MX,GLEVEL)	G3 02610
	G3 02620
**** ENTRY REDUNDANCY CHECK ****	G3 02630
S = 0	G3 02640
T = 0	G3 02650
CALL UNNECE	G3 02660
GATES = M	G3 02670
C = 0	G3 02680
DO 4 GI = 1,NR	G3 02690
C = C + LISUCC(GI)	G3 02700
IF(GI.LE.NM)GOTO4	G3 02710
IF(LISUCC(GI).GT.0)GATES=GATES+1	G3 02720
4 CONTINUE	G3 02730
OLDOST = A*GATES + B*(C)	G3 02740
T=0	G3 02750
S=0	G3 02760
INITIALIZE TIMER TO 10 MINUTES	G3 02770
CALL STIMEZ(60000)	G3 02780
TIME = KTIMEZ(0)	G3 02790
*** PROCEDURE GTMERG	G3 02800



CALL GTMERG	G3 02
CALL FOR ELAPSED TIME	G3 02
TIME = KTIMEZ(0) - TIME	G3 02
CALL LINE(4)	G3 02
PRINT 3915	G3 02
3916 FORMAT(20X,'TIME ELAPSED =',I8,' CENTISECONDS')	G3 02
3915 FORMAT(20X,'NETWORK DERIVED BY GTMERG')	G3 02
PRINT 3916,TIME	G3 02
CALL LINE(4)	G3 02
CALL TRUTH(P\$,1)	G3 02
CALL LINE(4)	G3 02
CALL CKT(INC\$MX,GLEVEL)	G3 02
GATES = M	G3 02
C = 0	G3 02
DO 36 GI = 1,NR	G3 02
C = C + LISUCC(GI)	G3 02
IF(GI.LE.NM) GO TO 36	G3 02
IF(LISUCC(GI).GT.0) GATES = GATES + 1	G3 02
36 CONTINUE	G3 02
NEWCST = A*GATES + B*C	G3 030
IF(NEWCST.LT.CLOCST)GO TO 37	G3 030
PRINT 105	G3 030
105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')	G3 030
GO TO 990	G3 030
37 CALL LINE(3)	G3 030
PRINT 320,NEWCST	G3 030
320 FORMAT(9X,'* A NETWORK DERIVED BY GTMERG'/9X,' COST=',I5,'.')	G3 030
GO TO 990	G3 030
500 STOP	G3 030
END	G3 031

SUBROUTINE MINI2(IMPROV)	G3 031
EDITION AA	G3 031
THE NAME ATTEMPTS TO INDICATE THAT THIS SUBROUTINE IS A MINIATURE	G3 031
VERSION OF PROCEDURE II (PROCII) - ACTUALLY, THIS ROUTINE ONLY	G3 031
REMOVES CONNECTIONS, NONE ARE ADDED	G3 031
IMPLICIT INTEGER*4(A-T,V-Z,\$), REAL(U)	G3 031

DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.

COMMON NEPMAX	G3 032
COMMON N , M , A , B	G3 032
1 , R , N2 , N1 , NR	G3 032
2 , NM , KFLAG , JFLAG , COST	G3 032
3 , LEVM , NRN2 , NM1 , NN2	G3 032
COMMON ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)	G3 032
1 , INC\$MX(40,40) , SUC\$MX(40,40) , P\$(2,1280) , UNAME(40)	G3 032
2 , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME	G3 032
COMMON T , RTCONN(100) , S , RSCONN(100)	G3 032
COMMON IFLAG , POINTA , ESS1S(40) , F\$1(32)	G3 032
1 , F\$UB1 , INPTCV(32) , LISTC(40) , POINTC	G3 033
2 , LISTL(40) , POINTL , ORIGIN(40) , IPATH(40)	G3 033
3 , POINTR , VF\$1(32) , VF\$UB1 , GSMALL(40,32)	G3 033
COMMON POTAB(200,42) , PPOTAB(40) , LPOTAB(40) , NRPLC(2)	G3 033
1 , RPLC(2,40) , IDX0(32) , IDX0E(32) , IDX1(32)	G3 033
2 , IDX1E(32) , SUMP(32) , SETT1(32) , NOT1	G3 033
3 , SETS1(40) , NOS1 , SETS(40) , NOS	G3 033
4 , STS , SUMS2(32) , SETS2(200) , NOS2	G3 033
5 , LIP , NOOE , KEYA , KEYB	G3 033
6 , NOO , NO1 , NO1E , \$GT	G3 033

7	, \$LTH	, \$PW	, \$NOE	, GI	G3 03400
	COMMON	NOT1SV	, NDS1SV	, LMTS2	G3 03410
	DIMENSION T1PRED(40), T2PRED(40), GORDER(40), F\$0(32), MARKED(40)				G3 03420
	DIMENSION USED(40), TORDER(40)				G3 03430
	IMPROV = 0				G3 03440
	T = 0				G3 03450
	ORDER GATES IN GORDER				G3 03460
	EFLAG = 0				G3 03470
	GO TO 63				G3 03480
	THIS ENTRY POINT FOR CALCULATION OF GORDER ONLY				G3 03490
	ENTRY FORMGO				G3 03500
	EFLAG = 1				G3 03510
63	CONTINUE				G3 03520
	COUNT = 0				G3 03530
	DO 1 I=1, LEVM				G3 03540
	NMINLV = LGLIST(I)				G3 03550
	IF(NMINLV.EQ.0)GOTO1				G3 03560
	DO 2 J=1, NMINLV				G3 03570
	COUNT = COUNT + 1				G3 03580
	GORDER(COUNT) = HLIST(J, I)				G3 03590
2	CONTINUE				G3 03600
1	CONTINUE				G3 03610
	IF(EFLAG.EQ.1)RETURN				G3 03620
	CALCULATE NUMBER OF OUTPUTS OF EACH GATE				G3 03630
	(THE ARRAY 'USED' IS USED HERE JUST TEMPORARILY)				G3 03640
	DO 51 I=N1, NR				G3 03650
	TCOUNT = 0				G3 03660
	DO 52 J=1, NR				G3 03670
	IF(INC\$MX(I, J).EQ.1)TCOUNT = TCOUNT + 1				G3 03680
52	CONTINUE				G3 03690
	TCOUNT NOW CONTAINS THE NUMBER OF OUTPUTS OF GATE I				G3 03700
	USED(I) = TCOUNT				G3 03710
51	CONTINUE				G3 03720
	MOST = 0				G3 03730
	DO 53 I =N1, NR				G3 03740
	IF(USED(I).GT.MOST)MOST = USED(I)				G3 03750
53	CONTINUE				G3 03760
	DO 56 I= 1, N				G3 03770
56	TORDER(I) = I				G3 03780
	TPoint = N1				G3 03790
	MOST = MOST + 1				G3 03800
50	MOST = MOST - 1				G3 03810
	IF(MOST.LT.0)GO TO 54				G3 03820
	DO 55 I=1, NR				G3 03830
	II = GORDER(I)				G3 03840
	IF(II.LE.N)GO TO 55				G3 03850
	IF(USED(II).NE.MOST)GO TO 55				G3 03860
	TORDER(TPOINT) = II				G3 03870
	TPoint = TPOINT + 1				G3 03880
55	CONTINUE				G3 03890
	GO TO 50				G3 03900
54	CONTINUE				G3 03910
	INITIALIZE GSMALL				G3 03920
	DO 4 I=N1, NM				G3 03930
	X = (I-1)*N2				G3 03940
	DO 4 J=1, N2				G3 03950
	Y = P\$(1, X+J)				G3 03960
	IF(Y.EQ.0)GSMALL(I, J) = -100				G3 03970
	IF(Y.EQ.1)GSMALL(I, J) = 1				G3 03980
	IF(Y.EQ.-1)GSMALL(I, J)=0				G3 03990
4	CONTINUE				G3 04000

EFLAG = 0	G3 040
GO TO 57	G3 040
ENTRY INITGS	G3 040
EFLAG = 1	G3 040
57 DO 3 I=1,NR	G3 040
USED(I) = 0	G3 040
IF(I.LT.N1)GO TO 58	G3 040
IF(I.GT.NM) GO TO 58	G3 040
GO TO 3	G3 040
58 DO 59 J = 1,N2	G3 041
59 GSMALL(I,J)= 0	G3 041
3 CONTINUE	G3 041
DO 62 I= N1,NM	G3 041
USED(I) = 1	G3 041
62 CONTINUE	G3 041
INITIALIZATION	G3 041
DO 34 I=1,NP	G3 041
GATE = GORDER(I)	G3 041
IF(GATE.LT.N1)GO TO 34	G3 041
XX= LIPRED(GATE)	G3 042
IF(XX.EQ.0)GOTO34	G3 042
F\$UB1 = 0	G3 042
F\$UB0 = 0	G3 042
DO 35 J=1,N2	G3 042
COMPNT = GSMALL(GATE,J)	G3 042
IF(COMPNT.EQ.0)GO TO 35	G3 042
IF(COMPNT.LT.0)GO TO 36	G3 042
IF(COMPNT.GE.1000) GO TO 35	G3 042
F\$UB0 = F\$UB0 + 1	G3 042
F\$0(F\$UB0) = J	G3 043
GO TO 35	G3 043
36 IF(COMPNT.LE.-1000) GO TO 35	G3 043
F\$UB1 = F\$UB1 + 1	G3 043
F\$1(F\$UB1) = J	G3 043
35 CONTINUE	G3 043
IF(F\$UB1.EQ.0)GO TO 34	G3 043
DO 38 K=1,XX	G3 043
FEEDGT = IPRED(K,GATE)	G3 043
X = (FEEDGT-1)*N2	G3 043
DO 39 L=1,F\$UB1	G3 044
Y = F\$1(L)	G3 044
IF(P\$(1,X+Y).LE.0)GO TO 39	G3 044
IF(GSMALL(FEEDGT,Y).GT.1000)GOTO39	G3 044
IF(GSMALL(GATE,Y).EQ.-200)GOTO39	G3 044
IF(GSMALL(GATE,Y).EQ.-100)GO TO 40	G3 044
GSMALL(GATE,Y) = -200	G3 044
GO TO 39	G3 044
40 GSMALL(GATE,Y) = -FEEDGT	G3 044
39 CONTINUE	G3 044
38 CONTINUE	G3 045
DO 60 K=1,XX	G3 045
60 MARKED(IPRED(K,GATE)) = 0	G3 045
DO 41 K=1,F\$UB1	G3 045
X = GSMALL(GATE,F\$1(K))	G3 045
IF(X.EQ.-100)GO TO 41	G3 045
IF(X.EQ.-200)GOTO41	G3 045
X = -X	G3 045
GSMALL(+X,F\$1(K))=1	G3 045
USED(X) = 1	G3 045
IF(MARKED(X).EQ.1)GOTO41	G3 046
MARKED(X) = 1	G3 046



DO 42 L=1,F\$UB0	G3 04620
Y = GSMALL(X,F\$0(L))	G3 04630
IF(Y.GT.1000.OR.Y.LT.-1000)GO TO 42	G3 04640
GSMALL(+X,F\$0(L))=-100	G3 04650
42 CONTINUE	G3 04660
41 CONTINUE	G3 04670
34 CONTINUE	G3 04680
IF(EFLAG.EQ.1)RETURN	G3 04690
INITIALIZE COUNTER TO LOOP ONCE FOR EACH GATE	G3 04700
G\$COUNT = 0	G3 04710
INCREMENT G\$COUNT	G3 04720
5 G\$COUNT = G\$COUNT + 1	G3 04730
ARE ALL GATES EXHAUSTED?	G3 04740
IF(G\$COUNT.LE.NR)GO TO 6	G3 04750
IF(T.GT.0) IMPROV = 1	G3 04760
IF(IMPROV.EQ.0)RETURN	G3 04770
IF HERE, NETWORK WAS ALTERED, SO UPDATE ARRAYS	G3 04780
CALL SUBNET	G3 04790
CALL PVALUE	G3 04800
RETURN	G3 04810
6 GCO = GORDER(G\$COUNT)	G3 04820
IS GCO AN ISOLATED GATE OF EXTERNAL VARIABLE?	G3 04830
IF(GCO.LE.N)GOTO5	G3 04840
DO 8 I=1,N2	G3 04850
IF(GSMALL(GCO,I).GE.1)GOTO7	G3 04860
8 CONTINUE	G3 04870
IF HERE, GATE IS ISOLATED - REMOVE INPUTS	G3 04880
X = LIPRED(GCO)	G3 04890
IF(X.EQ.0)GOTO5	G3 04900
DO 9 I=1,X	G3 04910
Y = IPPRED(I,GCO)	G3 04920
INC\$MX(Y,GCO) = 0	G3 04930
RECORD THE DISCONNECTION	G3 04940
T = T + 1	G3 04950
9 CONTINUE	G3 04960
GOTO 5	G3 04970
REMOVE UNNECESSARY CONNECTIONS TO GCO IN THE NEXT FEW SECTIONS	G3 04980
	G3 04990
CALCULATE F(GCO)	G3 05000
7 F\$UB1 = 0	G3 05010
DO 10 I=1,N2	G3 05020
IF(GSMALL(GCO,I).GE.0)GOTO10	G3 05030
F\$UB1 = F\$UB1 + 1	G3 05040
F\$1(F\$UB1) = I	G3 05050
10 CONTINUE	G3 05060
DO 11 I=1,F\$UB1	G3 05070
11 INPTCV(F\$1(I)) = 0	G3 05080
X = LIPRED(GCO)	G3 05090
DO 222 I=1,X	G3 05100
ESS1S(IPRED(I,GCO)) = 0	G3 05110
222 CONTINUE	G3 05120
T1SUB = 0	G3 05130
T2SUB = 0	G3 05140
DO 48 I = 1,NR	G3 05150
IF(INC\$MX(I,GCO).EQ.0)GOTO48	G3 05160
T1SUB = T1SUB + 1	G3 05170
T1PRED(T1SUB) = I	G3 05180
48 CONTINUE	G3 05190
17 DO 18 I=1,X	G3 05200
Y = (T1PRED(I)-1)*N2	G3 05210
DO 19 J=1,F\$UB1	G3 05220

Q = F\$1(J)	G3 05230
IF(P\$(1,Y+Q).NE.1)GO TO 19	G3 05240
IF(INPTCV(Q).LE.0) GO TO 20	G3 05250
INPTCV(Q) = INPTCV(Q) + 1	G3 05260
GO TO 19	G3 05270
20 IF(INPTCV(Q).LT.0)GO TO 21	G3 05280
INPTCV(Q) = -T1PRED(I)	G3 05290
GO TO 19	G3 05300
21 INPTCV(Q) = 2	G3 05310
19 CONTINUE	G3 05320
18 CONTINUE	G3 05330
MARK ESSENTIAL 1'S	G3 05340
DO 22 I=1,F\$UB1	G3 05350
Q = INPTCV(F\$1(I))	G3 05360
IF(Q.GE.0)GO TO 22	G3 05370
ESS1S(-Q) = ESS1S(-Q) + 1	G3 05380
22 CONTINUE	G3 05390
46 SELECT = 0	G3 05400
BESTSL = 0	G3 05410
DO 45 L=1,X	G3 05420
Q = T1PRED(L)	G3 05430
IF(INC\$MX(Q,GCO).EQ.0)GOTO45	G3 05440
IF(ESS1S(Q).GT.0)GOTO45	G3 05450
IF(SELECT.EQ.0)SELECT = Q	G3 05460
IF(USED(Q).EQ.1)GOTO45	G3 05470
IF(BESTSL.NE.0)GOTO45	G3 05480
BESTSL = Q	G3 05490
45 CONTINUE	G3 05500
IF(SELECT.EQ.0)GO TO 47	G3 05510
Q = SELECT	G3 05520
IF(BESTSL.NE.0)Q = BESTSL	G3 05530
IF HERE, GATE HAS NO ESSENTIAL 1'S - REMOVE IT	G3 05540
INC\$MX(Q,GCO) = 0	G3 05550
T = T + 1	G3 05560
UPDATE ESS1S	G3 05570
Y = (Q - 1)*N2	G3 05580
DO 24 J=1,F\$UB1	G3 05590
V = F\$1(J)	G3 05600
IF(P\$(1,Y+V).NE.1)GO TO 24	G3 05610
UPDATE INPTCV FOR COMPONENT V	G3 05620
INPTCV(V) = INPTCV(V) - 1	G3 05630
IF(INPTCV(V).GT.1)GO TO 24	G3 05640
CASE WHEN NEW ESSEN 1 CREATED	G3 05650
DO 27 K = 1,X	G3 05660
W = T1PRED(K)	G3 05670
IF(INC\$MX(W,GCO).EQ.0) GO TO 27	G3 05680
Z = (W - 1) * N2	G3 05690
IF(P\$(1,Z+V).EQ.0)GO TO 27	G3 05700
ESS1S(W) = ESS1S(W) + 1	G3 05710
IN THIS CASE, NO NEED TO UPDATE INPTCV FURTHER	G3 05720
GSMALL(GCO,V) = -W	G3 05730
GO TO 24	G3 05740
27 CONTINUE	G3 05750
24 CONTINUE	G3 05760
GOTO46	G3 05770
47 DO 49 I = 1,NR	G3 05780
IF(INC\$MX(I,GCO).EQ.0)GOTO49	G3 05790
T2SUB = T2SUB + 1	G3 05800
T2PRED(T2SUB) = I	G3 05810
49 CONTINUE	G3 05820
NOW ALL CURRENT INPUTS HAVE ESSENTIAL 1'S	G3 05830

INPUTS STILL CONNECTED TO GCO ARE LISTED IN T2PRED IN REVERSE  
ORDER

UPDATE G(I)'S OF THOSE GATES STILL CONNECTED TO GATE GCO

```

00 29 II=1,F$UB1
I = F$1(II)
CHOICE = -GSMALL(GCO,I)
IF(CHOICE.LT.100)GO TO 61
CHOICE = 0
00 30 JJJ=1,NR
JJ = TORDER(JJJ)
IF(INC$MX(JJ,GCO).EQ.0)GO TO 30
IF(P$(1,(JJ-1)*N2+I).NE.1)GO TO 30
IF(JJ.LE.N)GO TO 29
IF(CHOICE.EQ.0)CHOICE=JJ
IF(GSMALL(JJ,I).GE.1)GOTO29
30 CONTINUE
61 GSMALL(CHOICE,I) = 1
USED(CHOICE) = 1
29 CONTINUE
00 32 I=1,N2
IF(GSMALL(GCO,I).LT.1)GO TO 32
00 33 J=1,T2SUB
IF(GSMALL(T2PRED(J),I).EQ.0)GSMALL(T2PRED(J),I)=-100
33 CONTINUE
32 CONTINUE
GOTO5
END

```

G3 05840  
G3 05850  
G3 05860  
G3 05870  
G3 05880  
G3 05890  
G3 05900  
G3 05910  
G3 05920  
G3 05930  
G3 05940  
G3 05950  
G3 05960  
G3 05970  
G3 05980  
G3 05990  
G3 06000  
G3 06010  
G3 06020  
G3 06030  
G3 06040  
G3 06050  
G3 06060  
G3 06070  
G3 06080  
G3 06090  
G3 06100  
G3 06110  
G3 06120

SUBROUTINE GTMERG

IMPLICIT INTEGER\*4(A-T,V-Z), REAL(U)

\*\*\* THIS SUBROUTINE TRIES TO MERGE TWO GATES, THE INTERSECTION OF  
WHOSE CSPF'S IS NOT EMPTY, TO ONE WHICH IS INSIDE THE INTERSECTION  
OF THEIR CSPF'S \*\*\*\*\*

DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.

COMMON NEPMAX

COMMON	N	, M	, A	, B	G3 06130
1	, R	, N2	, N1	, NR	G3 06140
2	, NM	, KFLAG	, JFLAG	, COST	G3 06150
3	, LEVM	, NRN2	, NM1	, NN2	G3 06160
COMMON	ISUCC(40,40)	, LISUCC(40)	, IPRED(40,40)	, LIPRED(40)	G3 06170
1	, INC\$MX(40,40)	, SUC\$MX(40,40)	, P\$(2,1280)	, UNAME(40)	G3 06180
2	, GLEVEL(40)	, LGLIST(40)	, HLIST(40,40)	, TIME	G3 06190
COMMON	T	, RTCONN(100)	, S	, RSCONN(100)	G3 06200
COMMON	IFLAG	, POINTA	, ESS1S(40)	, F\$1(32)	G3 06210
1	, F\$UB1	, INPTCV(32)	, LISTC(40)	, POINTC	G3 06220
2	, LISTL(40)	, POINTL	, ORIGIN(40)	, IPATH(40)	G3 06230
3	, POINTR	, VF\$1(32)	, VF\$UB1	, GSMALL(40,32)	G3 06240
COMMON	POTAB(200,42)	, PPOTAB(40)	, LPOTAB(40)	, NRPLC(2)	G3 06250
1	, RPLC(2,40)	, IDX0(32)	, IDX0E(32)	, IDX1(32)	G3 06260
2	, IDX1E(32)	, SUMP(32)	, SETT1(32)	, NOT1	G3 06270
3	, SETS1(40)	, NOS1	, SETS(40)	, NOS	G3 06280
4	, STS	, SUMS2(32)	, SETS2(200)	, NOS2	G3 06290
5	, LIP	, NOOE	, KEYA	, KEYB	G3 06300
6	, NOO	, NO1	, NO1E	, \$GT	G3 06310
7	, \$LTH	, \$PW	, \$NOE	, GI\$\$\$\$	G3 06320
COMMON		NOT1SV	, NOS1SV	, LMTS2	G3 06330
					G3 06340
					G3 06350
					G3 06360
					G3 06370
					G3 06380
					G3 06390
					G3 06400
					G3 06410
					G3 06420

```

DIMENSION INDEX0(32),INDEX1(32),LICAND(40)
1 CONTINUE
CALL MINI2(IMPROV)
NR1=NR-1
DO 59 GI=NR1,NR1
  IF(GLEVEL(GI).EQ.1) GO TO 59
  BSGI=(GI-1)*N2
  GI1=GI+1
  DO 49 GJ=GI1,NR
    IF(GLEVEL(GJ).EQ.1 .OR. INC$MX(GI,GJ).GE.1
1    .OR. INC$MX(GJ,GI).GE.1) GO TO 49
    KGIGJ=0
    KGJGI=0
    NOO=0
    NO1=0
    BSGJ=(GJ-1)*N2
    DO 19 TH=1,N2
      IF(GSMALL(GI,TH)) 5,10,15
C***** TH OF GI IS A '0' *****
      5    IF(GSMALL(GJ,TH)) 6,7,8
C***** TH OF GI AND GJ ARE BOTH '0' *****
      6    NOO=NOO+1
      INDEX0(NOO)=TH
      GO TO 19
C***** TH OF GI IS A '0' AND TH OF GJ IS A '1' *****
      7    IF(P$(1,BSGJ+TH).EQ.0) GO TO 6
C***** GJ CAN NOT SUBSTITUTE FOR GI *****
      KGJGI=1
      GO TO 6
C***** GI, GJ CAN NOT BE MERGED TO ONE GATE *****
      8    GO TO 49
C
C***** GSMALL(GI,TH) IS A DON'T CARE *****
C
      10   IF(GSMALL(GJ,TH)) 11, 12, 13
C***** TH OF GJ IS A '0' *****
      11   IF(P$(1,BSGI+TH).EQ.0) GO TO 6
      KGIGJ=1
      GO TO 6
C***** BOTH GI AND GJ ARE DON'T CARES *****
      12   GO TO 19
C***** CORRESPONDING COMPONENT OF GJ IS '1' *****
      13   IF(P$(1,BSGI+TH).EQ.1) GO TO 18
      KGIGJ=1
      GO TO 18
C***** GSMALL(GI,TH) IS A '1' *****
      15   IF(GSMALL(GJ,TH)) 16, 17, 18
C***** GI AND GJ CAN NOT BE MERGED TO ONE GATE *****
      16   GO TO 49
C***** GSMALL(GJ,TH) IS A DON'T CARE *****
      17   IF(P$(1,BSGJ+TH).EQ.1) GO TO 18
C***** GJ CAN NOT SUBSTITUTE FOR GI *****
      KGJGI=1
C***** BOTH GSMALL(GI,TH) AND GSMALL(GJ,TH) ARE '1' *****
      18   NO1=NO1+1
      INDEX1(NO1)=TH
      19 CONTINUE
C
C***** GI AND GJ MAY BE MERGED *****
      IF (KGIGJ.EQ.0.AND.SUC$MX(GJ,GI).LE.0) GO TO 65
      IF(KGJGI.EQ.0 .AND. SUC$MX(GI,GJ).LE.0) GO TO 60

```

```

G3 06430
G3 06440
G3 06450
G3 06460
G3 06470
G3 06480
G3 06490
G3 06500
G3 06510
G3 06520
G3 06530
G3 06540
G3 06550
G3 06560
G3 06570
G3 06580
G3 06590
G3 06600
G3 06610
G3 06620
G3 06630
G3 06640
G3 06650
G3 06660
G3 06670
G3 06680
G3 06690
G3 06700
G3 06710
G3 06720
G3 06730
G3 06740
G3 06750
G3 06760
G3 06770
G3 06780
G3 06790
G3 06800
G3 06810
G3 06820
G3 06830
G3 06840
G3 06850
G3 06860
G3 06870
G3 06880
G3 06890
G3 06900
G3 06910
G3 06920
G3 06930
G3 06940
G3 06950
G3 06960
G3 06970
G3 06980
G3 06990
G3 07000
G3 07010
G3 07020
G3 07030

```



*****	TRY TO BUILD UP A NEW GATE TO REPLACE GI AND GJ *****	G3 07040
	CANDDT=0	G3 07050
	DO 29 GK=1,NR	G3 07060
	IF(GLEVEL(GK).EQ.1.AND.GK.GT.NM) GO TO 29	G3 07070
	IF(SUC\$MX(GI,GK).GT.0.OR.SUC\$MX(GJ,GK).GT.0) GO TO 29	G3 07080
*****	CHECK '1' COORDINATES(CORRESPONDING INPUT COORDINATE SHOULD BE 0)	G3 07090
	BSGK=(GK-1)*N2	G3 07100
*****	IF(N00.EQ.0) GO TO (GI,GJ ARE REDUNDANT)	G3 07110
*****	IF(N01.EQ.0) GO TO (GI,GJ AND GATES FED BY GI,GJ ARE REDUNDANT)	G3 07120
	DO 23 NRUN=1,N01	G3 07130
	IF(P\$(1,BSGK+INDEX1(NRUN)).EQ.1) GO TO 29	G3 07140
23	CONTINUE	G3 07150
*****	GK IS CONNECTIBLE TO THE NEW GATE *****	G3 07160
	CANDT=CANDDT+1	G3 07170
	LICAND(CANDT)=GK	G3 07180
29	CONTINUE	G3 07190
	IF(CANDDT.EQ.0) GO TO 49	G3 07200
	DO 30 NRUN=1,N00	G3 07210
30	SUMP(NRUN)=0	G3 07220
	DO 35 CAND=1,CANDDT	G3 07230
	GK=LICAND(CAND)	G3 07240
	BSGK=(GK-1)*N2	G3 07250
	DO 33 NRUN=1,N00	G3 07260
33	SUMP(NRUN)=SUMP(NRUN)+P\$(1,BSGK+INDEX0(NRUN))	G3 07270
35	CONTINUE	G3 07280
		G3 07290
*****	CHECK IF '0' COORDINATES ARE COVERED *****	G3 07300
		G3 07310
	DO 36 NRUN=1,N00	G3 07320
	IF(SUMP(NRUN).LE.0) GO TO 49	G3 07330
36	CONTINUE	G3 07340
*****	A NEW GATE CAN BE CONSTRUCTED TO REPLACE GI AND GJ *****	G3 07350
*****	MODIFY GI TO BE THE NEW GATE *****	G3 07360
70	LIP=LIPRED(GI)	G3 07370
	DO 80 LI=1,LIP	G3 07380
	T=T+1	G3 07390
	INC\$MX(IPRED(LI,GI),GI)=0	G3 07400
80	CONTINUE	G3 07410
	DO 81 CAND=1,CANDDT	G3 07420
	S=S+1	G3 07430
	INC\$MX(LICAND(CAND),GI)=1	G3 07440
81	CONTINUE	G3 07450
*****	CONNECT GI TO SUCCESSORS OF GJ *****	G3 07460
82	LIS=LISUCC(GJ)	G3 07470
	DO 85 LI=1,LIS	G3 07480
	GK=ISUCC(LI,GJ)	G3 07490
	T=T+1	G3 07500
	INC\$MX(GJ,GK)=0	G3 07510
	S=S+1	G3 07520
	INC\$MX(GI,GK)=1	G3 07530
85	CONTINUE	G3 07540
	CALL SUBNET	G3 07550
	CALL UNNECE	G3 07560
	CALL PVALUE	G3 07570
*****	'CALL RDTGNT' IS NOT NECESSARY SINCE 'MINI2' WILL DO IT *****	G3 07580
	GO TO 1	G3 07590
60	GK=GJ	G3 07600
	GJ=GI	G3 07610
	GI=GK	G3 07620
	GO TO 82	G3 07630
65	CONTINUE	G3 07640

GO TO 82	G3 07650
49 CONTINUE	G3 07660
59 CONTINUE	G3 07670
RETURN	G3 07680
END	G3 07690

SUBROUTINE SUBNET	G3 07700
IMPLICIT INTEGER*4(A-T,V-Z,\$), REAL(U)	G3 07710

DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.	G3 07730
	G3 07740

COMMON NEPMAX	G3 07750
COMMON     N                     , M                     , A                     , B	G3 07760
1     , R                     , N2                     , N1                     , NR	G3 07770
2     , NM                     , KFLAG                     , JFLAG                     , COST	G3 07780
3     , LEVM                     , NRN2                     , NM1                     , NN2	G3 07790
COMMON     ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)	G3 07800
1     , INC\$MX(40,40) , SUC\$MX(40,40) , P\$(2,1280) , UNAME(40)	G3 07810
2     , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME	G3 07820
COMMON     T                     , RTCONN(100) , S                     , RSCONN(100)	G3 07830
COMMON     IFLAG                     , POINTA                     , ESS1S(40)                     , F\$1(32)	G3 07840
1     , F\$UB1                     , INPTCV(32)                     , LISTC(40)                     , POINTC	G3 07850
2     , LISTL(40)                     , POINTL                     , ORIGIN(40)                     , IPATH(40)	G3 07860
3     , POINTR                     , VF\$1(32)                     , VF\$UB1                     , GSMALL(40,32)	G3 07870
COMMON     POTAB(200,42) , PPOTAB(40)                     , LPOTAB(40)                     , NRPLC(2)	G3 07880
1     , RPLC(2,40)                     , IDX0(32)                     , IDX0E(32)                     , IDX1(32)	G3 07890
2     , IDX1E(32)                     , SUMP(32)                     , SETT1(32)                     , NOT1	G3 07900
3     , SETS1(40)                     , NOS1                     , SETS(40)                     , NOS	G3 07910
4     , STS                     , SUMS2(32)                     , SETS2(200)                     , NOS2	G3 07920
5     , LIP                     , NOOE                     , KEYA                     , KEYB	G3 07930
6     , NOO                     , NO1                     , NO1E                     , \$GT	G3 07940
7     , \$LTH                     , \$PW                     , \$NCE                     , G\$11111	G3 07950
COMMON     NOT1SV                     , NOS1SV                     , LMTS2	G3 07960
DIMENSION X(40) , LX(40,2) , OUT0(40)	G3 07970
ENTRY PRESUC	G3 07980
1 CONTINUE	G3 07990
DO 10 GI=1,NR	G3 08000
LS=0	G3 08010
LP=0	G3 08020
DO 5 GJ=1,NR	G3 08030
IF(INC\$MX(GI,GJ).EQ.0) GO TO 3	G3 08040
LS=LS+1	G3 08050
ISUCC(LS,GI)=GJ	G3 08060
GO TO 5	G3 08070
3     IF(INC\$MX(GJ,GI).EQ.0) GO TO 5	G3 08080
LP=LP+1	G3 08090
IPRED(LP,GI)=GJ	G3 08100
5     CONTINUE	G3 08110
LISUCC(GI)=LS	G3 08120
LIPRED(GI)=LP	G3 08130
10 CONTINUE	G3 08140
ENTRY SUCCES	G3 08150
DO 21 GI=1,NR	G3 08160
DO 21 GJ=1,NR	G3 08170
SUC\$MX(GI,GJ)=0	G3 08180
21 CONTINUE	G3 08190
DO 30 GJ=N1,NR	G3 08200
DO 22 GS=1,NR	G3 08210
X(GS)=0	G3 08220
	G3 08230



22	CONTINUE	G3 08240
	X(GJ)=1	G3 08250
	LO=1	G3 08260
	LX(1,1)=GJ	G3 08270
	V=1	G3 08280
23	CONTINUE	G3 08290
	V=1-V	G3 08300
	SWO=1+V	G3 08310
	SW1=2-V	G3 08320
	L1=0	G3 08330
	DO 28 LL=1,LO	G3 08340
	GM=LX(LL,SWO)	G3 08350
	LIP=LIPRED(GM)	G3 08360
	IF(LIP.EQ.0) GO TO 28	G3 08370
	DO 26 LP=1,LIP	G3 08380
	GP=IPRED(LP,GM)	G3 08390
	IF(X(GP).GT.0) GO TO 26	G3 08400
	SUC\$MX(GP,GJ)=1	G3 08410
	L1=L1+1	G3 08420
	LX(L1,SW1)=GP	G3 08430
	X(GP)=1	G3 08440
26	CONTINUE	G3 08450
28	CONTINUE	G3 08460
	IF(L1.EQ.0) GO TO 30	G3 08470
	LO=L1	G3 08480
	GO TO 23	G3 08490
30	CONTINUE	G3 08500
	ENTRY LEVEL	G3 08510
	DO 40 GJ=1,NR	G3 08520
	DUTO(GJ)=LISUCC(GJ)	G3 08530
	GLEVEL(GJ)=-1	G3 08540
40	CONTINUE	G3 08550
	LEV=0	G3 08560
45	LEV=LEV+1	G3 08570
	G=0	G3 08580
	DO 50 GJ=1,NR	G3 08590
	IF(DUTO(GJ).GT.0 .OR. GLEVEL(GJ).GT.0) GO TO 50	G3 08600
	G=G+1	G3 08610
	HLIST(G,LEV)=GJ	G3 08620
	GLEVEL(GJ)=LEV	G3 08630
50	CONTINUE	G3 08640
	IF(G.EQ.0) RETURN	G3 08650
	LGLIST(LEV)=G	G3 08660
	DO 60 GG=1,G	G3 08670
	GJ=HLIST(GG,LEV)	G3 08680
	LIP=LIPRED(GJ)	G3 08690
	IF(LIP.EQ.0) GO TO 60	G3 08700
	DO 55 LP=1,LIP	G3 08710
	GP=IPRED(LP,GJ)	G3 08720
	DUTO(GP)=DUTO(GP)-1	G3 08730
55	CONTINUE	G3 08740
60	CONTINUE	G3 08750
	LEVM=LEV	G3 08760
	GO TO 45	G3 08770
		G3 08780
		G3 08790
		G3 08800
		G3 08810
	ENTRY PVALUE	G3 08820
	DO 100 L=NN2,NRN2	G3 08830
	P\$(1,L)=1	G3 08840

100	CONTINUE	G3	08850
	LEV=LEVM	G3	08860
110	CONTINUE	G3	08870
	LD=LGLIST(LEV)	G3	08880
	DO 130 L=1,LD	G3	08890
	GI=HLIST(L,LEV)	G3	08900
	LIS=LISUCC(GI)	G3	08910
	BSGI=(GI-1)*N2	G3	08920
	LJTH=0	G3	08930
	DO 115 JTH=1,N2	G3	08940
	IF(P\$(1,BSGI+JTH).EQ.0) GO TO 115	G3	08950
	LJTH=LJTH+1	G3	08960
	X(LJTH)=JTH	G3	08970
115	CONTINUE	G3	08980
	IF(LJTH.EQ.0) GO TO 130	G3	08990
	DO 125 LS=1,LIS	G3	09000
	GS=ISUCC(LS,GI)	G3	09010
	BSGS=(GS-1)*N2	G3	09020
	DO 120 LJ=1,LJTH	G3	09030
	P\$(1,X(LJ)+BSGS)=0	G3	09040
120	CONTINUE	G3	09050
125	CONTINUE	G3	09060
130	CONTINUE	G3	09070
	LEV=LEV-1	G3	09080
	IF(LEV.GE.2) GO TO 110	G3	09090
	RETURN	G3	09100
		G3	09110
		G3	09120
		G3	09130
		G3	09140
	ENTRY RSTRCT(KEYRST)	G3	09150
	KEYRST=0	G3	09160
	IF(LEVM.GT.LMAX)GO TO 160	G3	09170
	DO 150 GI=N1,NR	G3	09180
	IF(LIPRED(GI).GT.FANIN)GO TO 160	G3	09190
	IF(LISUCC(GI).GT.FANOUT)GO TO 160	G3	09200
150	CONTINUE	G3	09210
	RETURN	G3	09220
160	KEYRST=1	G3	09230
	RETURN	G3	09240
	ENTRY UNNECE	G3	09250
C*****	THIS ENTRY DISCONNECT ALL GATES FROM WHICH THERE IS NO PATH	G3	09260
C	TO OUTPUT GATES *****	G3	09270
	TS=T	G3	09280
	DO 209 GI=NM1,NR	G3	09290
	IF(GLEVEL(GI).EQ.1) GO TO 207	G3	09300
	DO 205 GJ=N1,NM	G3	09310
	IF(SUC\$MX(GI,GJ).GT.0) GO TO 209	G3	09320
205	CONTINUE	G3	09330
C*****	GI IS REDUNDANT *****	G3	09340
207	CONTINUE	G3	09350
	LIP=LIPRED(GI)	G3	09360
	IF(LIP.EQ.0) GO TO 206	G3	09370
	DO 203 LI=1,LIP	G3	09380
	GK=IPRED(LI,GI)	G3	09390
	IF(INC\$MX(GK,GI).LE.0) GO TO 203	G3	09400
	T=T+1	G3	09410
	RTCONN(T)=100*GK+GI	G3	09420
	INC\$MX(GK,GI)=0	G3	09430
203	CONTINUE	G3	09440
206	LIS=LISUCC(GI)	G3	09450

IF(LIS.EQ.0) GO TO 209	G3 09460
DO 204 LI=1,LIS	G3 09470
GK=ISUCC(LI,GI)	G3 09480
IF(INC\$MX(GI,GK).LE.0) GO TO 204	G3 09490
T=T+1	G3 09500
RTCONN(T)=100*GI+GK	G3 09510
INC\$MX(GI,GK)=0	G3 09520
204 CONTINUE	G3 09530
209 CONTINUE	G3 09540
IF(T.GT.TS) GO TO 1	G3 09550
RETURN	G3 09560
END	G3 09570
SUBROUTINE OUTPUT(MATRIX,ARRAY)	G3 09580
IMPLICIT INTEGER*4(A-T,V-Z,\$), REAL(U)	G3 09590
	G3 09600
DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.	G3 09610
	G3 09620
COMMON NEPMAX	G3 09630
COMMON N, M, A, B	G3 09640
1, R, N2, N1, NR	G3 09650
2, NM, KFLAG, JFLAG, COST	G3 09660
3, LEVM, NRN2, NM1, NN2	G3 09670
COMMON ISUCC(40,40), LISUCC(40), IPRED(40,40), LIPRED(40)	G3 09680
1, INC\$MX(40,40), SUC\$MX(40,40), P\$(2,1280), UNAME(40)	G3 09690
2, GLEVEL(40), LGLIST(40), HLIST(40,40), TIME	G3 09700
COMMON T, RTCONN(100), S, RSCONN(100)	G3 09710
COMMON IFLAG, POINTA, ESS1S(40), F\$1(32)	G3 09720
1, F\$UB1, INPTCV(32), LISTC(40), POINTC	G3 09730
2, LISTL(40), POINTL, ORIGIN(40), IPATH(40)	G3 09740
3, POINTR, VF\$1(32), VF\$UB1, GSMALL(40,32)	G3 09750
COMMON POTAB(200,42), PPOTAB(40), LPOTAB(40), NRPLC(2)	G3 09760
1, RPLC(2,40), IDX0(32), IDX0E(32), IDX1(32)	G3 09770
2, IDX1E(32), SUMP(32), SETT1(32), NOT1	G3 09780
3, SETS1(40), NOS1, SETS(40), NOS	G3 09790
4, STS, SUMS2(32), SETS2(200), NOS2	G3 09800
5, LIP, NOOE, KEYA, KEYB	G3 09810
6, NCO, NO1, NO1E, \$GT	G3 09820
7, \$LTH, \$PW, \$NOE, GI	G3 09830
COMMON NOT1SV, NOS1SV, LMTS2	G3 09840
DIMENSION UX(5), UY(5), UG(40), UF(40), ARRAY(40), ARRAY2(2,1280)	G3 09850
DIMENSION MATRIX(40,40)	G3 09860
DATA UX /' X1', ' X2', ' X3', ' X4', ' X5' /	G3 09870
DATA UY /' Y1', ' Y2', ' Y3', ' Y4', ' Y5' /	G3 09880
DATA UF /' 1', ' 2', ' 3', ' 4', ' 5', ' 6', ' 7', ' 8'	G3 09890
1, ' 9', ' 10', ' 11', ' 12', ' 13', ' 14', ' 15', ' 16'	G3 09900
2, ' 17', ' 18', ' 19', ' 20', ' 21', ' 22', ' 23', ' 24'	G3 09910
3, ' 25', ' 26', ' 27', ' 28', ' 29', ' 30', ' 31', ' 32'	G3 09920
4, ' 33', ' 34', ' 35', ' 36', ' 37', ' 38', ' 39', ' 40' /	G3 09930
DATA GMAX/40/	G3 09940
	G3 09950
KEYXC=ARRAY(1)	G3 09960
IF(KEYXC.NE.0) GO TO 50	G3 09970
DO 1 GI=1,N	G3 09980
UNAME(GI)=UX(GI)	G3 09990
1 CONTINUE	G3 10000
GO TO 100	G3 10010
50 CONTINUE	G3 10020
L=N/2	G3 10030
DO 4 GI=1,L	G3 10040

UNAME(GI)=UX(GI)	G3 10050
UNAME(GI+L)=UY(GI)	G3 10060
4 CONTINUE	G3 10070
100 CONTINUE	G3 10080
DO 2 GI=N1,GMAX	G3 10090
UNAME(GI)=UF(GI-N)	G3 10100
2 CONTINUE	G3 10110
RETURN	G3 10120
ENTRY LINE(L)	G3 10130
DO 6 LL=1,L	G3 10140
PRINT 5	G3 10150
5 FORMAT(1H )	G3 10160
6 CONTINUE	G3 10170
RETURN	G3 10180
ENTRY PAGE	G3 10190
PRINT 7	G3 10200
7 FORMAT(1H1)	G3 10210
RETURN	G3 10220
ENTRY CKT(MATRIX,ARRAY)	G3 10230
PRINT 10	G3 10240
10 FORMAT(1H ,8X,'GATE .. LEVEL',6X, 'FED BY'//)	G3 10250
DO 20 GJ=N1,NR	G3 10260
G=0	G3 10270
DO 15 GI=1,NR	G3 10280
IF(MATRIX(GI,GJ).EQ.0) GO TO 15	G3 10290
G=G+1	G3 10300
UG(G)=UNAME(GI)	G3 10310
15 CONTINUE	G3 10320
IF(G.EQ.0) GO TO 18	G3 10330
PRINT 17, UNAME(GJ),ARRAY(GJ),(UG(GG),GG=1,G)	G3 10340
17 FORMAT(1H0, 9X,A3,5X,'/',I2,'/',5X,35( A3))	G3 10350
GO TO 20	G3 10360
18 PRINT 19, UNAME(GJ),ARRAY(GJ)	G3 10370
19 FORMAT(1H0, 9X,A3,5X,'/',I2,'/')	G3 10380
20 CONTINUE	G3 10390
RETURN	G3 10400
ENTRY TRUTH(ARRAY2,J)	G3 10410
IF(J.EQ.2) GO TO 36	G3 10420
PRINT 35	G3 10430
35 FORMAT(11X, 'TRUTH TABLE'//)	G3 10440
GO TO 38	G3 10450
36 PRINT 37	G3 10460
37 FORMAT(11X, 'REQUIREMENT TABLE')	G3 10470
38 CONTINUE	G3 10480
DO 40 GI=1,NR	G3 10490
ILO=(GI-1)*N2+1	G3 10500
IHI=ILO+N2-1	G3 10510
PRINT 41, UNAME(GI), (ARRAY2(J,I),I=ILO,IHI)	G3 10520
40 CONTINUE	G3 10530
41 FORMAT(1H0, 9X,A3,' = ', 32(I1,1X))	G3 10540
RETURN	G3 10550
END	G3 10560
	G3 10570
	G3 10580
	G3 10590
	G3 10600



```

*****
PPPP   RRRR   ODD   GGG   RRRR   A   M   M
P  P   R  R   O  O   G  G   R  R   A  A   MM  MM
P  P   R  R   O  O   G  G   R  R   A  A   M M M M
PPPP   RRRR   O  O   G  GG   RRRR   AAAAA  M  M  M
P      R  R   O  O   G  G   R  R   A  A   M    M
P      R  R   ODD   GGG   R  R   A  A   M    M
*****

```

```

N  V   EEEEE   TTTT   TTTT   RRRR   A   GGG   4
NN  N   E       T       T       R  R   A  A   G  G   44
N  N  N   E       T       T       R  R   A  A   G       4 4
N  NN   EEE       T       T       RRRR   AAAAA  XXXXX  G  GG   4 4
N  N   E       T       T       R  R   A  A   G  G   44444
N  N   EEEEE   T       T       R  R   A  A   GGG       4
*****

```

```

IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                                G4 00010
EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBG4 00020
NOTE: ALL COMMON VARIABLES MIGHT NOT BE USED IN THIS PROGRAM.        G4 00030
                                                                           G4 00040

```

#### COMMON VARIABLES:

```

$GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY          G4 00060
     IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.                    G4 00070
$LTH: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY          G4 00080
     IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.            G4 00090
$NOE: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'POW' THE ENTRY          G4 00100
     IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS        G4 00110
     ROW IS USED.                                                        G4 00120
$PW: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY          G4 00130
     IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.                        G4 00140
A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.                 G4 00150
B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.           G4 00160
COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.                     G4 00170
ESSIS: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCOG4 00180
      (POSITIONS IN ESSIS CORRES. TO GATES NOT FEEDING GCO ARE        G4 00190
      IGNORED).                                                         G4 00200
F$UB1: POINTS TO LAST ELEMENT IN F$1.                                  G4 00210
F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR             G4 00220
     COVERING) IN A CONNECTIBLE FUNCTION.                              G4 00230
GI: LABEL OF A PARTICULAR GATE.                                         G4 00240
GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.          G4 00250
GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.               G4 00260
HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-      G4 00270
      WORK LEVEL J.                                                     G4 00280
IDX0: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER                G4 00290
      CONSIDERATION.                                                    G4 00300
IDX0E: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER         G4 00310
      CONSIDERATION.                                                    G4 00320
IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER                G4 00330
      CONSIDERATION.                                                    G4 00340
IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER        G4 00350

```

CONSIDERATION.	G4 00360
IFLAG: SAME AS EYFLG IN SUBROUTINE PROCII.	G4 00370
INC\$MX: INC\$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE	G4 00380
(OR EX. VAR.) GI TO GATE GJ. INC\$MX(GI,GJ)=0 IF NOT.	G4 00390
INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F\$1, HOW MANY INPUTS	G4 00400
HAVE A '1' IN THE POSITION INDICATED BY F\$1.	G4 00410
IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATE	G4 00420
TO AN OUTPUT GATE. OTHERWISE IPATH(GI) = 0.	G4 00430
IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. IN	G4 00440
A LIST OF GATES AND EX. VAR. FEEDING GJ.	G4 00450
ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.	G4 00460
JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.	G4 00470
KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN	G4 00480
PERFORMED.	G4 00490
KEYB: A FLAG INDICATING IF ANY PRIMARY O-ERROR-COORDINATES HAS	G4 00500
BEEN COMPENSATED.	G4 00510
KFLAG: SAME AS KEIFLG IN PROCII.	G4 00520
LEV: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO	G4 00530
ASSIGNED LEVELS JUST LIKE GATES).	G4 00540
LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF	G4 00550
NETWORK.	G4 00560
LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.	G4 00570
LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.	G4 00580
LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO. ORDERED BY	G4 00590
DECREASING NO. OF O'S IN GCO COVERED.	G4 00600
LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED	G4 00610
GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED. ORDERED BY	G4 00620
DECREASING NO. OF ESSENTIAL I'S.	G4 00630
LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR	G4 00640
EX. VAR.) GI.	G4 00650
LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.	G4 00660
LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB	G4 00670
CONCERNING GI.	G4 00680
M: NUMBER OF NETWORK OUTPUT GATES.	G4 00690
N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.	G4 00700
NEPMAX: FOR ERROR COMPENSATION PROGRAMS. IF MORE THAN NEPMAX	G4 00710
ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,	G4 00720
PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S	G4 00730
REMOVAL. VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL	G4 00740
TO ONE HALF OF N2 BY DEFAULT.	G4 00750
NM: SUM OF N PLUS M	G4 00760
NM1: SUM OF NM PLUS 1.	G4 00770
NN2: PRODUCT OF N AND N2.	G4 00780
NOS: NUMBER OF ELEMENTS IN SET S.	G4 00790
NOS1: NUMBER OF ELEMENTS IN SET S1.	G4 00800
NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE	G4 00810
RPLCF.	G4 00820
NOS2: NUMBER OF ELEMENTS IN SET S2.	G4 00830
NOT1: NUMBER OF ELEMENTS IN SET T1.	G4 00840
NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE	G4 00850
RPLCF.	G4 00860
NOO: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDXO.	G4 00870
NOOE: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDXOE.	G4 00880
NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.	G4 00890
NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.	G4 00900
NR: SUM OF N PLUS R.	G4 00910
NFN2: PRODUCT OF NR AND N2.	G4 00920
NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)	G4 00930
FOR I=1,2.	G4 00940
N1: SUM OF N PLUS 1.	G4 00950
N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED	G4 00960



(USUALLY 2 TO THE POWER N).

ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO. G4 00970  
ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY. G4 00980  
P\$: P\$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND G4 00990  
EVERY GATE (FOR EVERY INPUT COMBINATION): P\$(1,1),..., G4 01000  
P\$(1,N2) FOR FIRST EX VAR; P\$(1,N2+1),...,P\$(1,2\*N2) FOR G4 01010  
SECOND EX VAR; ... ; P\$(1,N\*N2+1),..., P\$(1,N\*N2+N2) FOR G4 01020  
FIRST GATE; ETC. P\$(2,-) IS USED AS WORK SPACE FOR G4 01030  
CALCULATIONS ASSOCIATED WITH P\$(1,-). G4 01040  
PCO: FOR ERROR COMPENSATION PROCEDURES. PCO IS THE GATE G4 01050  
REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED G4 01060  
NETWORK. G4 01070  
POINTA: NOT USED. G4 01080  
POINTC: POINTS TO LAST ELEMENT IN LISTC. G4 01090  
POINTL: POINTS TO LAST ELEMENT IN LISTL. G4 01100  
POINTR: POINTS TO LAST ELEMENT IN RNECI (IN SUBROUTINE SUBSTI). G4 01110  
POTAB: POSSIBLE OUTPUT TABLE. HOLDS INFORMATION ABOUT ALL G4 01120  
COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY G4 01130  
USEFUL) FUNCTIONS. G4 01140  
PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF G4 01150  
ROWS OF POTAB CONCERNING GI. G4 01160  
R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO G4 01170  
NOTE SOME OF R GATES MAY BE ISOLATED). G4 01180  
RPLC: RPLC(1,\*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE G4 01190  
ERROR-COORDINATES OF WEIGHT 2 OR ABOVE. G4 01200  
RPLC(2,\*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE G4 01210  
AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1. G4 01220  
RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM). G4 01230  
RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM). G4 01240  
S: NO. OF CONNECTIONS ADDED TO A NETWORK. POINTS TO LAST G4 01250  
ENTRY IN RSCONN. G4 01260  
SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATION G4 01270  
WHICH ARE TO BE REPLACED IF POSSIBLE. G4 01280  
SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE G4 01290  
REPLACED BY ELEMENTS IN SET S2. G4 01300  
SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR G4 01310  
REPLACING ELEMENTS IN SET S. G4 01320  
SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS IN G4 01330  
SET S1. G4 01340  
STS: STARTING ELEMENT OF SET S. G4 01350  
SUC\$MX: SUC\$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI. G4 01360  
SUC\$MX(GI,GJ)=0 IF NOT. G4 01370  
SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION. G4 01380  
SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2. G4 01390  
T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK. POINTS TO G4 01400  
LAST ENTRY IN RTCONN. G4 01410  
TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME. G4 01420  
UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES. G4 01430  
VFSUB1: POINTS TO LAST ELEMENT IN VFS1. G4 01440  
VFS1: SIMILAR TO F\$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONS G4 01450  
(OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING G4 01460  
ORIGINALLY CONNECTED INPUTS TO GCO. G4 01470  
G4 01480  
G4 01490  
G4 01500  
G4 01510  
G4 01520  
G4 01530  
G4 01540  
G4 01550  
G4 01560  
G4 01570

COMMON NEPMAX  
COMMON N , M , A , B  
1 , R , N2 , N1 , NR  
2 , NM , KFLAG , JFLAG , COST  
3 , LEVM , NRN2 , NM1 , NN2  
COMMON ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)

```

1      , INC$MX(40,40), SUC$MX(40,40), P$(2,1280) , UNAME(40) G4 01580
2      , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME G4 01590
COMMON T , RTCONN(100) , S , RSCONN(100) G4 01600
COMMON IFLAG , POINTA , ESSIS(40) , F$1(32) G4 01610
1      , F$UB1 , INPTCV(32) , LISTC(40) , POINTC G4 01620
2      , LISTL(40) , POINTL , ORIGIN(40) , IPATH(40) G4 01630
3      , POINTR , VF$1(32) , VF$UB1 , GSMALL(40,32) G4 01640
COMMON PDATAB(200,42), PPOTAB(40) , LPOTAB(40) , NRPLC(2) G4 01650
1      , RPLC(2,40) , IDX0(32) , IDX0E(32) , IDX1(32) G4 01660
2      , IDX1E(32) , SUMP(32) , SETT1(32) , NOT1 G4 01670
3      , SETS1(40) , NOS1 , SETS(40) , NOS G4 01680
4      , STS , SUMS2(32) , SETS2(200) , NOS2 G4 01690
5      , LIP , NOOE , KEYA , KEYB G4 01700
6      , NOO , NO1 , NO1E , $GT G4 01710
7      , $LTH , $PW , $NOE , GI G4 01720
COMMON NOT1SV , NOS1SV , LMTS2 G4 01730
DIMENSION CNTLIS(144),UGATE(40),UHEAD(20) G4 01740
DATA KOUNT5 /0/, UBLANK/' ' G4 01750
990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX G4 01760
NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS G4 01770
1000 FORMAT(20A4/5I4,A4,I4) G4 01780
KEYXC=0 G4 01790
IF(UC.NE.UBLANK) KEYXC=1 G4 01800
CALL PAGE G4 01810
CALL LINE(10) G4 01820
KOUNT5=KOUNT5+1 G4 01830
PRINT 2, KOUNT5 G4 01840
2 FORMAT(20X,'*** OPTIMAL NCR NETWORK ***',50X,'PROBLEM NO.= ',I4 ) G4 01850
CALL LINE(4) G4 01860
PRINT 1005, UHEAD G4 01870
1005 FORMAT(25X,20A4) G4 01880
CALL LINE(4) G4 01890
PRINT 10, N,M,A,B G4 01900
10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 // G4 01910
1 30X,'NUMBER OF FUNCTIONS =',I4 // G4 01920
2 30X,'COST COEFFICIENT A =',I4// G4 01930
3 47X, 'B =',I4) G4 01940
CALL LINE(1) G4 01950
IF(KEYXC.NE.0) GO TO 25 G4 01960
PRINT 21 G4 01970
21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES X ---') G4 01980
GO TO 30 G4 01990
25 CONTINUE G4 02000
PRINT 28 G4 02010
28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLES G4 02020
1 X, Y ---') G4 02030
30 CONTINUE G4 02040
CALL LINE(5) G4 02050
***** SET UP EXTERNAL VARIABLES ***** G4 02060
N2=2**N G4 02070
IF(NEPMAX.EQ.0)NEPMAX = N2/2 G4 02080
H=N*N2 G4 02090
J=N2 G4 02100
L= 1 G4 02110
I=0 G4 02120
DO 1011 II=1,N G4 02130
J=J/2 G4 02140
L=L*2 G4 02150
SN= 1 G4 02160
DO 1010 LL=1,L G4 02170
SN=-SN G4 02180

```

V=(1+SN)/2	G4 02190
DC 1009 JJ=1,J	G4 02200
I=I+1	G4 02210
P\$(1,I)=V	G4 02220
IF(KEYXC.NE.0)P\$(1,I+H)=1-V	G4 02230
1009 CONTINUE	G4 02240
1010 CONTINUE	G4 02250
1011 CONTINUE	G4 02260
IF(KEYXC.NE.0) N=N+N	G4 02270
N1=N+1	G4 02280
NM=N+M	G4 02290
NM1=NM+1	G4 02300
NN2=N*N2+1	G4 02310
NR=N+R	G4 02320
NRN2=NP*N2	G4 02330
CALL OUTPUT(INC\$MX,KEYXC)	G4 02340
C***** READ IN NETWORK INFORMATION AND SET UP INC\$MX *****	G4 02350
READ 1001, CNTLIS	G4 02360
1001 FORMAT(16I5)	G4 02370
DO 1115 GI=1,NR	G4 02380
DO 1115 GJ=1,NR	G4 02390
1115 INC\$MX(GI,GJ)=0	G4 02400
DO 1120 I=1,144	G4 02410
ITEM=CNTLIS(I)	G4 02420
IF(ITEM.EQ.0) GO TO 1119	G4 02430
GI=ITEM/100	G4 02440
GJ=ITEM-100*GI	G4 02450
INC\$MX(GI,GJ)=1	G4 02460
GO TO 1120	G4 02470
1119 COST=A*R+B*(I-1)	G4 02480
GO TO 1130	G4 02490
1120 CONTINUE	G4 02500
1130 CONTINUE	G4 02510
CALL SUBNET	G4 02520
CALL PVALUE	G4 02530
CALL LINE(4)	G4 02540
PRINT 1140, COST	G4 02550
1140 FORMAT(20X,' ORIGINAL NETWORK COST=', I5)	G4 02560
CALL LINE(4)	G4 02570
CALL TRUTH(P\$,1)	G4 02580
CALL LINE(4)	G4 02590
CALL CKT(INC\$MX,GLEVEL)	G4 02600
C	G4 02610
C***** ENTRY REDUNDANCY CHECK *****	G4 02620
S = 0	G4 02630
T = 0	G4 02640
CALL UNNECE	G4 02650
GATES = M	G4 02660
C = 0	G4 02670
DO 4 GI = 1,NR	G4 02680
C = C + LISUCC(GI)	G4 02690
IF(GI.LE.NM)GOTO4	G4 02700
IF(LISUCC(GI).GT.0)GATES=GATES+1	G4 02710
4 CONTINUE	G4 02720
OLD COST = A*GATES + B*(C)	G4 02730
T=0	G4 02740
S=0	G4 02750
C INITIALIZE TIMER TO 10 MINUTES	G4 02760
CALL STIMEZ(60000)	G4 02770
TIME = KTIMEZ(0)	G4 02780
C***** PROCEDURE PROCV	G4 02790



CALL PROCV	G4 02800
CALL FOR ELAPSED TIME	G4 02810
TIME = KTIMEZ(0) - TIME	G4 02820
CALL LINE(4)	G4 02830
PRINT 3915	G4 02840
3916 FORMAT(20X,'TIME ELAPSED =',I8,' CENTISECONDS')	G4 02850
3915 FORMAT(20X,'NETWORK DERIVED BY PROCV')	G4 02860
PRINT 3916,TIME	G4 02870
CALL LINE(4)	G4 02880
CALL TRUTH(P\$,1)	G4 02890
CALL LINE(4)	G4 02900
CALL CKT(INC\$MX,GLEVEL)	G4 02910
GATES = M	G4 02920
C = 0	G4 02930
DO 36 GI = 1,NR	G4 02940
C = C + LISUCC(GI)	G4 02950
IF(GI.LE.NM) GO TO 36	G4 02960
IF(LISUCC(GI).GT.0) GATES = GATES + 1	G4 02970
36 CONTINUE	G4 02980
NEWGST = A*GATES + B*C	G4 02990
IF(NEWGST.LT.OLDCST)GO TO 37	G4 03000
PRINT 105	G4 03010
105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')	G4 03020
GO TO 990	G4 03030
37 CALL LINE(3)	G4 03040
PRINT 320,NEWGST	G4 03050
320 FFORMAT(9X,'* A NETWORK DERIVED BY PROCV '/9X,' COST=',I5,'.')	G4 03060
GO TO 990	G4 03070
500 STOP	G4 03080
END	G4 03090

SUBROUTINE OUTPUT(MATRIX,ARRAY)	G4 03100
IMPLICIT INTEGER*4(A-T,V-Z,\$), REAL(U)	G4 03110

DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.	G4 03130
---	----------

COMMON NEPMAX	G4 03150
COMMON N , M , A , B	G4 03160
1 , R , N2 , N1 , NR	G4 03170
2 , NM , KFLAG , JFLAG , COST	G4 03180
3 , LEVM , NRN2 , NM1 , NN2	G4 03190
COMMON ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)	G4 03200
1 , INC\$MX(40,40) , SUC\$MX(40,40) , P\$(2,1280) , UNAME(40)	G4 03210
2 , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME	G4 03220
COMMON T , RTCONN(100) , S , RSCONN(100)	G4 03230
COMMON IFLAG , POINTA , ESS1S(40) , F\$1(32)	G4 03240
1 , F\$UB1 , INPTCV(32) , LISTC(40) , POINTC	G4 03250
2 , LISTL(40) , POINTL , ORIGIN(40) , IPATH(40)	G4 03260
3 , POINTR , VF\$1(32) , VF\$UB1 , GSMALL(40,32)	G4 03270
COMMON POTAB(200,42) , PPOTAB(40) , LPOTAB(40) , NRPLC(2)	G4 03280
1 , RPLC(2,40) , IDX0(32) , IDX0E(32) , IDX1(32)	G4 03290
2 , IDX1E(32) , SUMP(32) , SETT1(32) , NCT1	G4 03300
3 , SETS1(40) , NOS1 , SETS(40) , NOS	G4 03310
4 , STS , SUMS2(32) , SETS2(200) , NOS2	G4 03320
5 , LIP , NOOE , KEYA , KEYB	G4 03330
6 , NOO , NO1 , NO1E , \$GT	G4 03340
7 , \$LTH , \$PW , \$NOE , GI	G4 03350
COMMON NOT1SV , NOS1SV , LMTS2	G4 03360
DIMENSION UX(5) , UY(5) , UG(40) , UF(40) , ARRAY(40) , ARRAY2(2,1280)	G4 03370
DIMENSION MATRIX(40,40)	G4 03380

DATA UX /' X1', ' X2', ' X3', ' X4', ' X5' /	G4 03390
DATA UY /' Y1', ' Y2', ' Y3', ' Y4', ' Y5' /	G4 03400
DATA UF /' 1', ' 2', ' 3', ' 4', ' 5', ' 6', ' 7', ' 8'	G4 03410
1 , ' 9', ' 10', ' 11', ' 12', ' 13', ' 14', ' 15', ' 16'	G4 03420
2 , ' 17', ' 18', ' 19', ' 20', ' 21', ' 22', ' 23', ' 24'	G4 03430
3 , ' 25', ' 26', ' 27', ' 28', ' 29', ' 30', ' 31', ' 32'	G4 03440
4 , ' 33', ' 34', ' 35', ' 36', ' 37', ' 38', ' 39', ' 40' /	G4 03450
DATA GMAX/40/	G4 03460
KEYXC=ARRAY(1)	G4 03470
IF(KEYXC.NE.0) GO TO 50	G4 03480
DO 1 GI=1,N	G4 03490
UNAME(GI)=UX(GI)	G4 03500
1 CONTINUE	G4 03510
GO TO 100	G4 03520
50 CONTINUE	G4 03530
L=N/2	G4 03540
DO 4 GI=1,L	G4 03550
UNAME(GI)=UX(GI)	G4 03560
UNAME(GI+L)=UY(GI)	G4 03570
4 CONTINUE	G4 03580
100 CONTINUE	G4 03590
DO 2 GI=N1,GMAX	G4 03600
UNAME(GI)=UF(GI-N)	G4 03610
2 CONTINUE	G4 03620
RETURN	G4 03630
	G4 03640
ENTRY LINE(L)	G4 03650
DO 6 LL=1,L	G4 03660
PRINT 5	G4 03670
5 FORMAT(1H )	G4 03680
6 CONTINUE	G4 03690
RETURN	G4 03700
	G4 03710
	G4 03720
ENTRY PAGE	G4 03730
PRINT 7	G4 03740
7 FORMAT(1H1)	G4 03750
RETURN	G4 03760
	G4 03770
ENTRY CKT(MATRIX,ARRAY)	G4 03780
PRINT 10	G4 03790
10 FORMAT(1H ,8X,'GATE .. LEVEL',6X, 'FED BY' /)	G4 03800
DO 20 GJ=N1,NR	G4 03810
G=0	G4 03820
DO 15 GI=1,NR	G4 03830
IF(MATRIX(GI,GJ).EQ.0) GO TO 15	G4 03840
G=G+1	G4 03850
UG(G)=UNAME(GI)	G4 03860
15 CONTINUE	G4 03870
IF(G.EQ.0) GO TO 18	G4 03880
PRINT 17, UNAME(GJ),ARRAY(GJ),(UG(GG),GG=1,G)	G4 03890
17 FORMAT(1H0, 9X,A3,5X,'/',I2,'/',5X,35( A3))	G4 03900
GO TO 20	G4 03910
18 PRINT 19, UNAME(GJ),ARRAY(GJ)	G4 03920
19 FORMAT(1H0, 9X,A3,5X,'/',I2,'/')	G4 03930
20 CONTINUE	G4 03940
RETURN	G4 03950
	G4 03960
ENTRY TRUTH(ARRAY2,J)	G4 03970
IF(J.EQ.2) GO TO 36	G4 03980
PRINT 35	G4 03990



```

35 FORMAT(11X, 'TRUTH TABLE'/)
GO TO 38
36 PRINT 37
37 FORMAT(11X, 'REQUIREMENT TABLE')
38 CONTINUE
DO 40 GI=1,NR
  ILO=(GI-1)*N2+1
  IHI=ILO+N2-1
  PRINT 41, UNAME(GI), (ARRAY2(J,I),I=ILO,IHI)
40 CONTINUE
41 FORMAT(1H0, 9X,A3,' = ', 32(11,1X))
RETURN
END

```

```

SUBROUTINE PROCV
**** THIS SUBROUTINE HAS AN ENTRY POINT PROCVS(GI) BESIDE PROCV ITSELF
WHEN THIS SUBROUTINE IS ENTERED FROM PROCV, PROCEDURE V IS APPLIED
TO ALL GATES IN THE NETWORK REPEATEDLY UNTIL NO GATE CAN BE
REMOVED BY THIS PROCEDURE. WHEN THIS SUBROUTINE IS ENTERED AT
ENTRY POINT PROCVS(GI), ONLY GATE GI IS EXAMINED TO SEE WHETHER IT
CAN BE REMOVED BY PROCEDURE V OR NOT *****
IMPLICIT INTEGER*4(A-T,V-Z), REAL(U)

DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.

COMMON NEPMAX
COMMON      N      , M      , A      , B
1      , R      , N2      , N1      , NR
2      , NM      , KFLAG      , JFLAG      , COST
3      , LEVM      , NRN2      , NM1      , NN2
COMMON      ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)
1      , INC$MX(40,40) , SUC$MX(40,40) , P$(2,1280) , UNAME(40)
2      , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME
COMMON      T      , RTCONN(100) , S      , RSCONN(100)
COMMON      IFLAG      , POINTA      , ESS1S(40) , F$1(32)
1      , F$UB1      , INPTCV(32) , LISTC(40) , POINTC
2      , LISTL(40) , POINTL      , ORIGIN(40) , IPATH(40)
3      , POINTR      , VF$1(32) , VF$UB1      , GSMALL(40,32)
COMMON      POTAB(200,42) , PPOTAB(40) , LPOTAB(40) , NRPLC(2)
1      , RPLC(2,40) , IDX0(32) , IDX0E(32) , IDX1(32)
2      , IDX1E(32) , SUMP(32) , SETT1(32) , NOT1
3      , SETS1(40) , NOS1      , SETS(40) , NOS
4      , STS      , SUMS2(32) , SETS2(200) , NOS2
5      , LIP      , NOOE      , KEYA      , KEYB
6      , NOO      , NO1      , NO1E      , $GT
7      , $LTH      , $PW      , $NOE      , GI$$$$
COMMON      NOT1SV      , NOS1SV      , LMTS2
COMMON      /VRQRNW/      GIGJ      , PGIGJ(1280) , LISTIJ(40)
DIMENSION INDEX0(32), INDEX1(32), LISCND(40), ISCND(40,40),
1      CNCTBL(40), NCTBL(40,40), TEMPRS(90), INDEXS(32)
ETRKEY = 0
***** SELECT GI WHICH IS GOING TO BE REMOVED *****
1 TS = T
  GI=NM
2 GI = GI + 1
  IF(GI.LE.NR) GO TO 3
  IF(T.EQ.TS) RETURN
  GO TO 1
***** ACTUAL ENTRY POINT FOR PROCVS(GI) *****
3 CONTINUE

```

IF(GLEVEL(GI).EQ.1)GO TO 199	G4 04590
***** SKIP GATES WHICH HAVE ONLY ONE OUTPUT TO GI *****	G4 04600
LIP=LIPRED(GI)	G4 04610
DO 10 LI=1,LIP	G4 04620
GJ=IPRED(LI,GI)	G4 04630
IF(GJ.LE.N)GO TO 10	G4 04640
IF(LISUCC(GJ).EQ.1.AND.ISUCC(1,GJ).GT.NM)GO TO 199	G4 04650
10 CONTINUE	G4 04660
C***** GATES WHICH HAVE ONLY ONE INPUT FROM GI ARE REDUNDANT *****	G4 04670
LIS=LISUCC(GI)	G4 04680
DO 11 LI=1,LIS	G4 04690
GJ=ISUCC(LI,GI)	G4 04700
IF(LISUCC(GJ).EQ.0)GO TO 11	G4 04710
IF(GJ.GT.NM.AND.LIPRED(GJ).EQ.1) GO TO 12	G4 04720
11 CONTINUE	G4 04730
GO TO 20	G4 04740
C***** GJ IS REDUNDANT. DISCONNECT CONNECTION GI TO GJ AND CONNECT ALL	G4 04750
INPUTS OF GI TO EVERY SUCCEEDING GATE OF GJ *****	G4 04760
12 LIS=LISUCC(GJ)	G4 04770
T=T+1	G4 04780
RTCONN(T)=100*GI+GJ	G4 04790
INC\$MX(GI,GJ)=0	G4 04800
DO 15 LI=1,LIS	G4 04810
GK=ISUCC(LI,GJ)	G4 04820
T=T+1	G4 04830
RTCONN(T)=100*GJ+GK	G4 04840
INC\$MX(GJ,GK)=0	G4 04850
DO 13 IP=1,LIP	G4 04860
GH=IPRED(IP,GI)	G4 04870
IF(INC\$MX(GH,GK).GE.1)GO TO 13	G4 04880
S=S+1	G4 04890
RSCONN(S)=100*GH+GK	G4 04900
INC\$MX(GH,GK)=1	G4 04910
13 CONTINUE	G4 04920
15 CONTINUE	G4 04930
CALL SUBNET	G4 04940
CALL UNNECE	G4 04950
GO TO 199	G4 04960
C***** CALL RQPNW TO CALCULATE REQUIREMENTS FOR EACH GATE *****	G4 04970
20 TEMPS=0	G4 04980
CALL RORNW(GI)	G4 04990
C***** SELECT GATES TO REPLACE CONNECTIONS FROM GI *****	G4 05000
DO 169 GIG=1,GIGJ	G4 05010
GJ=LISTIJ(GIG)	G4 05020
B\$GIGJ=(GIG-1)*N2	G4 05030
C***** LIST "0" AND "1" COMPONENTS IN REQUIRED FUNCTION OF GI TO GJ *****	G4 05040
N01=0	G4 05050
N00=0	G4 05060
DO 52 TH=1,N2	G4 05070
IF(PGIGJ(B\$GIGJ+TH))52,50,51	G4 05080
50 N00=N00+1	G4 05090
INDEX0(N00)=TH	G4 05100
GO TO 52	G4 05110
51 N01=N01+1	G4 05120
INDEX1(N01)=TH	G4 05130
52 CONTINUE	G4 05140
IF(N01.EQ.0) GO TO 169	G4 05150
	G4 05160
	G4 05170
	G4 05180
	G4 05190

NO11=NO1

```
C ***** PARTITION GATES NOT FED BY GI INTO THREE CATAGORIES:
C 1.NCTBL: CONNECTIBLE GATES
C 2.ISCND: MAY BE MADE CONNECTIBLE GATES
C 3.OTHERS *****
  CNDKNT=0
  CBLKNT=0
  DO 56 GK=1,NR
    IF(GK.EQ.GI) GO TO 56
    IF(GK.GT.NM.AND.GLEVEL(GK).EQ.1) GO TO 56
    IF(SUC$MX(GI,GK).GE.1)GO TO 56
    BSGK=(GK-1)*N2

C ***** COMPARE GK WITH PGIGJ *****
  EFSIGN=0
C ***** EFSIGN=1: EFFECTIVE; EFSIGN=0: NON-EFFECTIVE *****
  DO 53 NRUN=1,NO1
    IF(P$(1,BSGK+INDEX1(NRUN)).EQ.0)GO TO 53
    EFSIGN=1
53  CONTINUE
    IF(EFSIGN.EQ.0)GO TO 56

  EFSIGN=0
C ***** EFSIGN=0: CONNECTABLE; EFSIGN=1: CANDIDATE *****
  DO 54 NRUN=1,NO0
    IF(P$(1,BSGK+INDEX0(NRUN)).EQ.0)GO TO 54
    IF(P$(2,BSGK+INDEX0(NRUN)).EQ.1)GO TO 56
    EFSIGN=1
54  CONTINUE
    IF(EFSIGN.EQ.0)GO TO 55
    IF(GK.LE.NM)GO TO 56
    CNDKNT=CNDKNT+1
    ISCND(CNDKNT,GIG)=GK
    GO TO 56
55  CBLKNT=CBLKNT+1
    NCTBL(CBLKNT,GIG)=GK
56  CONTINUE
    LISCND(GIG)=CNDKNT
    CNCTBL(GIG)=CBLKNT

C ***** SELECT ESSENTIAL GATES FROM NCTBL *****
  CNCT=CNCTBL(GIG)
  IF(CNCT.EQ.0)GO TO 62
  DO 59 CNC=1,CNCT
    EFSIGN=0
    GK=NCTBL(CNC,GIG)
    BSGK=(GK-1)*N2
    DO 57 NRUN=1,NO1
      IF(INDEX1(NRUN).GT.100)GO TO 57
      TH=INDEX1(NRUN)
      IF(P$(1,BSGK+TH).NE.1)GO TO 57
      INDEX1(NRUN)=100+TH
      P$(2,BSGK+TH)=1
      EFSIGN=1
57  CONTINUE
      IF(EFSIGN.EQ.0)GO TO 59
      TEMPS=TEMPS+1
      TEMPRS(TEMPS)=100*GK+GJ
      INC$MX(GK,GJ)=1
      CALL SUBNET
```

G4 05200  
G4 05210  
G4 05220  
G4 05230  
G4 05240  
G4 05250  
G4 05260  
G4 05270  
G4 05280  
G4 05290  
G4 05300  
G4 05310  
G4 05320  
G4 05330  
G4 05340  
G4 05350  
G4 05360  
G4 05370  
G4 05380  
G4 05390  
G4 05400  
G4 05410  
G4 05420  
G4 05430  
G4 05440  
G4 05450  
G4 05460  
G4 05470  
G4 05480  
G4 05490  
G4 05500  
G4 05510  
G4 05520  
G4 05530  
G4 05540  
G4 05550  
G4 05560  
G4 05570  
G4 05580  
G4 05590  
G4 05600  
G4 05610  
G4 05620  
G4 05630  
G4 05640  
G4 05650  
G4 05660  
G4 05670  
G4 05680  
G4 05690  
G4 05700  
G4 05710  
G4 05720  
G4 05730  
G4 05740  
G4 05750  
G4 05760  
G4 05770  
G4 05780  
G4 05790  
G4 05800



C		G4 05810
C*****	UPDATE REQUIREMENTS FOR GK *****	G4 05820
	IF(GK.LE.NM)GO TO 59	G4 05830
	DO 58 NRUN=1,NOO	G4 05840
58	P\$(2,BSGK+INDEXO(NRUN))=0	G4 05850
	CALL RQRGT(GK)	G4 05860
59	CONTINUE	G4 05870
C		G4 05880
C*****	COMPRESS THE TABLE OF UNCOVERED ONE COMPONENTS *****	G4 05890
	NO11=0	G4 05900
	DO 61 NRUN=1,NO1	G4 05910
	IF(INDEX1(NRUN).GT.100)GO TO 61	G4 05920
	NO11=NO11+1	G4 05930
	INDEX1(NO11)=INDEX1(NRUN)	G4 05940
61	CONTINUE	G4 05950
C		G4 05960
C*****	GATES CONNECTED TO GJ CAN REPLACE CONNECTION GI TO GJ(NO11=0)*****	G4 05970
	IF(NO11.EQ.0)GO TO 169	G4 05980
C		G4 05990
C*****	FIND CANDIDATES FOR EACH UNCOVERED ONE *****	G4 06000
C		G4 06010
62	LISCN=LISCND(GIG)	G4 06020
	IF(LISCN.EQ.0)GO TO 185	G4 06030
	DO 168 LISC=1,LISCN	G4 06040
	GK=LISCND(LISC,GIG)	G4 06050
	BSGK=(GK-1)*N2	G4 06060
	DO 63 NRUN=1,NO11	G4 06070
	TTH=INDEX1(NRUN)	G4 06080
	IF(P\$(1,BSGK+TTH).EQ.1)GO TO 64	G4 06090
63	CONTINUE	G4 06100
	GO TO 168	G4 06110
C		G4 06120
C*****	CHECK WHICH COMPONENT OF GK HAS TO BE CHANGED *****	G4 06130
C		G4 06140
64	NO10=0	G4 06150
	DO 65 NRUN=1,NOO	G4 06160
	IF(P\$(1,BSGK+INDEXO(NRUN)).NE.1)GO TO 65	G4 06170
	IF(P\$(2,BSGK+INDEXO(NRUN)).GE.0)GO TO 168	G4 06180
	NO10=NO10+1	G4 06190
	INDEXS(NO10)=INDEXO(NRUN)	G4 06200
65	CONTINUE	G4 06210
C		G4 06220
C*****	FIND A FUNCTION GL, CONNECTING WHICH TO GK WILL MAKE GK	G4 06230
C	CONNECTIBLE TO GJ *****	G4 06240
	DO 90 GL=1,NR	G4 06250
	IF(GL.GT.NM.AND.GLEVEL(GL).EQ.1) GO TO 90	G4 06260
	IF(SUC\$MX(GI,GL).GE.1.OR.SUC\$MX(GK,GL).GE.1) GO TO 90	G4 06270
C		G4 06280
C*****	COMPARE FUNCTION GL WITH REQUIREMENTS FOR GJ *****	G4 06290
	BSGL=(GL-1)*N2	G4 06300
	IF(P\$(1,BSGL+TTH).NE.0)GO TO 90	G4 06310
	DO 67 NRUN=1,NO10	G4 06320
	TH=INDEXS(NRUN)	G4 06330
	IF(P\$(1,BSGL+TH).NE.1)GO TO 90	G4 06340
67	CONTINUE	G4 06350
	DO 69 TH=1,N2	G4 06360
	IF(P\$(2,BSGK+TH))69,69,68	G4 06370
68	IF(P\$(1,BSGL+TH).EQ.1)GO TO 90	G4 06380
69	CONTINUE	G4 06390
C		G4 06400
C*****	CONNECT GL TO GK *****	G4 06410

```

TEMPS=TEMPS+1
TEMPS(TEMPS)=100*GL+GK
INC$MX(GL,GK)=1
TEMPS=TEMPS+1
TEMPS(TEMPS)=100*GK+GJ
INC$MX(GK,GJ)=1
CALL SUBNET

```

```

C***** UPDATE TRUTH TABLE FOR GK *****

```

```

DO 71 TH=1,N2
  IF(P$(1,B$GL+TH))71,71,70
70   P$(1,B$GK+TH)=0
71   CONTINUE
  CALL UPTRTH(GI, GK)

```

```

C***** UPDATE REQUIREMENT TABLE FOR GK *****

```

```

DO 72 NRUN=1,NCO
  P$(2,B$GK+INDEX0(NRUN))=0
72   CONTINUE
  NO1=NO11
  NO11=0
  DO 74 NRUN=1,NO1
    TH=INDEX1(NRUN)
    IF(P$(1,B$GK+TH).EQ.1)GO TO 73
    NO11=NO11+1
    INDEX1(NO11)=TH
    GO TO 74
73   P$(2,B$GK+TH)=1
74   CONTINUE
    CALL RQRGT(GK)
    IF(NO11.EQ.0)GO TO 169
  GO TO 168
90   CONTINUE
168  CONTINUE
  GO TO 185
169  CONTINUE

```

```

C***** GI CAN BE ELIMINATED *****

```

```

C***** LIST ALL ADDED CONNECTIONS *****

```

```

IF(TEMPS.LE.0) GO TO 173
DO 172 TEM=1,TEMPS
  S=S+1
  RSCONN(S)=TEMPS(TEM)
172 CONTINUE

```

```

C***** DISCONNECT ALL CONNECTIONS RELATED TO GI *****

```

```

173 CONTINUE
  LIS=LISUCC(GI)
  DO 175 LI=1,LIS
    GJ=ISUCC(LI,GI)
    T=T+1
    RTCONN(T)=100*GI+GJ
    INC$MX(GI,GJ)=0
175 CONTINUE
  LIP=LIPRED(GI)
  DO 176 LI=1,LIP
    GJ=IPRED(LI,GI)
    T=T+1
    RTCONN(T)=100*GJ+GI
    INC$MX(GJ,GI)=0

```

```

G4 06420
G4 06430
G4 06440
G4 06450
G4 06460
G4 06470
G4 06480
G4 06490
G4 06500
G4 06510
G4 06520
G4 06530
G4 06540
G4 06550
G4 06560
G4 06570
G4 06580
G4 06590
G4 06600
G4 06610
G4 06620
G4 06630
G4 06640
G4 06650
G4 06660
G4 06670
G4 06680
G4 06690
G4 06700
G4 06710
G4 06720
G4 06730
G4 06740
G4 06750
G4 06760
G4 06770
G4 06780
G4 06790
G4 06800
G4 06810
G4 06820
G4 06830
G4 06840
G4 06850
G4 06860
G4 06870
G4 06880
G4 06890
G4 06900
G4 06910
G4 06920
G4 06930
G4 06940
G4 06950
G4 06960
G4 06970
G4 06980
G4 06990
G4 07000
G4 07010
G4 07020

```



176	CONTINUE	G4	07030
	CALL SUBNET	G4	07040
	CALL UNNECE	G4	07050
	CALL PVALUE	G4	07060
	GO TO 199	G4	07070
C		G4	07080
C*****	GI CAN NOT BE ELIMINATED. RESTORE THE ORINGINAL NETWORK *****	G4	07090
185	IF(TEMPS.EQ.0)GO TO 199	G4	07100
	DO 189 TEM=1,TEMPS	G4	07110
	GL=TEMPRS(TEM)/100	G4	07120
	GK=TEMPRS(TEM)-GL*100	G4	07130
	IF(GL.GT.N.OR.GK.GT.NM) GO TO 188	G4	07140
	S=S+1	G4	07150
	RSCONN(S)=TEMPRS(TEM)	G4	07160
	GO TO 189	G4	07170
188	CONTINUE	G4	07180
	TNC\$MX(GL,GK)=0	G4	07190
189	CONTINUE	G4	07200
	CALL SUBNET	G4	07210
	CALL PVALUE	G4	07220
199	CONTINUE	G4	07230
	IF(ETRKEY.EQ.0) GO TO 2	G4	07240
	RETURN	G4	07250
	ENTRY PROCVS(GI)	G4	07260
	ETRKEY=1	G4	07270
	GO TO 3	G4	07280
	END	G4	07290
	SUBROUTINE RQRNW(GI)	G4	07300
	IMPLICIT INTEGER*4(A-T,V-Z), REAL(U)	G4	07310
C		G4	07320
C		G4	07330
C	DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.	G4	07340
		G4	07350
	COMMON NEPMAX		
	COMMON N , M , A , B	G4	07360
1	, R , N2 , N1 , NR	G4	07370
2	, NM , KFLAG , JFLAG , COST	G4	07380
3	, LEVM , NRN2 , NM1 , NN2	G4	07390
	COMMON ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)	G4	07400
1	, INC\$MX(40,40) , SUC\$MX(40,40) , P\$(2,1280) , UNAME(40)	G4	07410
2	, GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME	G4	07420
	COMMON T , RTCONN(100) , S , RSCONN(100)	G4	07430
	COMMON IFLAG , POINTA , ESS1S(40) , F\$1(32)	G4	07440
1	, F\$UB1 , INPTCV(32) , LISTC(40) , POINTC	G4	07450
2	, LISTL(40) , POINTL , ORIGIN(40) , IPATH(40)	G4	07460
3	, POINTR , VF\$1(32) , VF\$UB1 , GSMALL(40,32)	G4	07470
	COMMON PDTAB(200,42) , PPDTAB(40) , LPDTAB(40) , NRPLC(2)	G4	07480
1	, RPLC(2,40) , IDX0(32) , IDX0E(32) , IDX1(32)	G4	07490
2	, IDX1E(32) , SUMP(32) , SETT1(32) , NOT1	G4	07500
3	, SETS1(40) , NOS1 , SETS(40) , NOS	G4	07510
4	, STS , SUMS2(32) , SETS2(200) , NOS2	G4	07520
5	, LIP , NOOE , KEYA , KEYB	G4	07530
6	, NOO , NOI , NOIE , \$GT	G4	07540
7	, \$LTH , \$PW , \$NOE , GI\$\$\$\$	G4	07550
	COMMON NOT1SV , NOS1SV , LMTS2	G4	07560
	COMMON /VRQRNW/ GIGJ , PGIGJ(1280) , LISTIJ(40)	G4	07570
	DIMENSION INDEX0(32), INDEX1(32), PSUM(32)	G4	07580
C		G4	07590
C*****	COPY TRUTH TABLE FOR OUTPUT GATES AND EXTERNAL VARIABLES *****	G4	07600
	MTH=NM*N2	G4	07610

DO 10 TH=1,MTH	G4 07620
10 P\$(2,TH)=P\$(1,TH)	G4 07630
C***** INITIALIZE ALL FUNCTIONS TO BE UNDEFINED *****	G4 07640
MTH1=MTH+1	G4 07650
MTH2=NR*N2	G4 07660
DO 11 TH=MTH1,MTH2	G4 07670
11 P\$(2,TH)=-1	G4 07680
C***** REQUIREMENTS FOR CONNECTIONS TO A CERTAIN LEVEL *****	G4 07690
GIGJ=0	G4 07700
LEVR=LEVM-2	G4 07710
DO 49 LEV=1,LEVR	G4 07720
LGL=LGLIST(LEV)	G4 07730
DO 45 LG=1,LGL	G4 07740
GJ=HLIST(LG,LEV)	G4 07750
IF(GJ.EQ.GI)GO TO 45	G4 07760
IF(GJ.GT.NM.AND.LEV.EQ.1) GO TO 45	G4 07770
IF(GJ.LE.N) GO TO 45	G4 07780
BSGJ=(GJ-1)*N2	G4 07790
LIP=LIPRED(GJ)	G4 07800
KEYGI=0	G4 07810
DO 25 TH=1,N2	G4 07820
25 PSUM(TH)=0	G4 07830
DO 35 LP=1,LIP	G4 07840
GK=IPRED(LP,GJ)	G4 07850
BSGK=(GK-1)*N2	G4 07860
IF(GK.NE.GI) GO TO 26	G4 07870
KEYGI=1	G4 07880
GO TO 35	G4 07890
26 IF(GK.GT.NM) GO TO 29	G4 07900
C***** GK IS EXTERNAL VARIABLE OR OUTPUT GATE, UPDATE PSUM ONLY *****	G4 07910
DO 28 TH=1,N2	G4 07920
IF(P\$(2,BSGK+TH).EQ.1)PSUM(TH)=PSUM(TH)+1	G4 07930
28 CONTINUE	G4 07940
GO TO 35	G4 07950
C***** REQUIEMENTS FOR CONNECTION GK TO GJ *****	G4 07960
29 DO 33 TH=1,N2	G4 07970
IF(P\$(2,BSGJ+TH)) 33,30,31	G4 07980
30 IF(PSUM(TH).GE.1.OR.P\$(1,BSGK+TH).NE.1) GO TO 33	G4 07990
P\$(2,BSGK+TH)=1	G4 08000
PSUM(TH)=1	G4 08010
GO TO 33	G4 08020
31 P\$(2,BSGK+TH)=0	G4 08030
33 CONTINUE	G4 08040
35 CONTINUE	G4 08050
IF(KEYGI.NE.1) GO TO 45	G4 08060
C***** CALCULATE REQUIREMENTS FOR CONNECTION GI TO GJ *****	G4 08070
GIGJ=GIGJ+1	G4 08080
BSGIGJ=(GIGJ-1)*N2	G4 08090
LISTIJ(GIGJ)=GJ	G4 08100
DO 39 TH=1,N2	G4 08110
IF(P\$(2,BSGJ+TH)) 38,36,37	G4 08120
36 IF(PSUM(TH).GE.1) GO TO 38	G4 08130
PGIGJ(BSGIGJ+TH)=1	G4 08140
GO TO 39	G4 08150
37 PGIGJ(BSGIGJ+TH)=0	G4 08160
GO TO 39	G4 08170
38 PGIGJ(BSGIGJ+TH)=-1	G4 08180
	G4 08190
	G4 08200
	G4 08210
	G4 08220

39	CONTINUE	G4 08230
45	CONTINUE	G4 08240
49	CONTINUE	G4 08250
	RETURN	G4 08260
		G4 08270
	***** UPDATE REQUIREMENTS FOR GATES FEEDING GZ *****	G4 08280
	ENTRY RQRGT(GZ)	G4 08290
	LEV1=GLEVEL(GZ)	G4 08300
	LEV2=LEVM-2	G4 08310
	IF(LEV2.LT.LEV1)RETURN	G4 08320
	DO 189 LEV=LEV1,LEV2	G4 08330
	KEYCHG=0	G4 08340
	LGL=LGLIST(LEV)	G4 08350
	DO 179 LG=1,LGL	G4 08360
	GY=HLIST(LG,LEV)	G4 08370
	IF(GY.LE.N) GO TO 179	G4 08380
	IF(SUC\$MX(GY,GZ).LE.0.AND.GY.NE.GZ)GO TO 179	G4 08390
	BSGY=(GY-1)*N2	G4 08400
	LIP=LIPRED(GY)	G4 08410
	N00=0	G4 08420
	N01=0	G4 08430
	DO 109 TH=1,N2	G4 08440
	IF(P\$(2,BSGY+TH)) 109,107,108	G4 08450
107	N00=N00+1	G4 08460
	INDEX0(N00)=TH	G4 08470
	GO TO 109	G4 08480
108	N01=N01+1	G4 08490
	INDEX1(N01)=TH	G4 08500
109	CONTINUE	G4 08510
	DO 110 TH=1,N00	G4 08520
110	PSUM(TH)=0	G4 08530
	DO 119 LI=1,LIP	G4 08540
	GX=IPRED(LI,GY)	G4 08550
		G4 08560
	BSGX=(GX-1)*N2	G4 08570
	DO 112 NRUN=1,N00	G4 08580
	IF(P\$(2,BSGX+INDEX0(NRUN))) 112, 112, 111	G4 08590
111	PSUM(NRUN)=PSUM(NRUN)+1	G4 08600
112	CONTINUE	G4 08610
119	CONTINUE	G4 08620
	N000=0	G4 08630
	N011=0	G4 08640
	DO 125 NRUN=1,N00	G4 08650
	IF(PSUM(NRUN).GE.1) GO TO 125	G4 08660
	N000=N000+1	G4 08670
	INDEX0(N000)=INDEX0(NRUN)	G4 08680
125	CONTINUE	G4 08690
	DO 133 LI=1,LIP	G4 08700
	GX=IPRED(LI,GY)	G4 08710
	IF(GX.LE.NM) GO TO 133	G4 08720
	BSGX=(GX-1)*N2	G4 08730
	IF(N000.EQ.0) GO TO 131	G4 08740
	DO 130 NRUN=1,N000	G4 08750
	TH=INDEX0(NRUN)	G4 08760
	IF(TH.GT.100) GO TO 130	G4 08770
	IF(P\$(1,BSGX+TH).NE.1) GO TO 130	G4 08780
	P\$(2,BSGX+TH)=1	G4 08790
	INDEX0(NRUN)=100+TH	G4 08800
130	CONTINUE	G4 08810
131	DO 132 NRUN=1,N01	G4 08820
	TH=INDEX1(NRUN)	G4 08830

	IF(P\$(2,BSGX+TH).NE.-1) GO TO 132	G4 08840
	P\$(2,BSGX+TH)=0	G4 08850
	NO11=NO11+1	G4 08860
132	CONTINUE	G4 08870
133	CONTINUE	G4 08880
	IF(NDOO.NE.0.OR.NO11.NE.0) KEYCHG=1	G4 08890
179	CONTINUE	G4 08900
	IF(KEYCHG.EQ.0) RETURN	G4 08910
189	CONTINUE	G4 08920
	RETURN	G4 08930
	ENTRY UPTRTH(GI,GZ)	G4 08940
****	WHEN THIS ENTRY POINT IS CALLED IT WILL UPDATE THE TRUTH TABLE OF	G4 08950
	GATES WHICH ARE SUCCESSORS OF GZ BUT NOT SUCCESSORS OF GI ****	G4 08960
	LEVZ=GLEVEL(GZ)	G4 08970
	DO 300 LEV=2,LEVZ	G4 08980
	VEL=LEVZ-LEV+1	G4 08990
	LGL=LGLIST(VEL)	G4 09000
	DO 290 LG=1,LGL	G4 09010
	GP=HLIST(LG,VEL)	G4 09020
	IF(GP.EQ.GI	G4 09030
1	.OR. SUC\$MX(GI,GP).GE.1	G4 09040
2	.OR. SUC\$MX(GZ,GP).LE.0	G4 09050
3	.OR. LIPRED(GP).LE.0) GO TO 290	G4 09060
	BSGP=(GP-1)*N2	G4 09070
	DO 210 TH=1,N2	G4 09080
	P\$(1,BSGP+TH)=1	G4 09090
	PSUM(TH)=0	G4 09100
210	CONTINUE	G4 09110
	LIP=LIPRED(GP)	G4 09120
	DO 260 LP=1,LIP	G4 09130
	GR=IPRED(LP,GP)	G4 09140
	BSGR=(GR-1)*N2	G4 09150
	DO 230 TH=1,N2	G4 09160
	PSUM(TH)=PSUM(TH)+P\$(1,BSGR+TH)	G4 09170
230	CONTINUE	G4 09180
260	CONTINUE	G4 09190
	DO 280 TH=1,N2	G4 09200
	IF(PSUM(TH).GE.1) P\$(1,BSGP+TH)=0	G4 09210
280	CONTINUE	G4 09220
290	CONTINUE	G4 09230
300	CONTINUE	G4 09240
	RETURN	G4 09250
	END	G4 09260

SUBROUTINE SUBNET	G4 09270
IMPLICIT INTEGER*4(A-T,V-Z,\$), REAL(U)	G4 09280

DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.	G4 09300
	G4 09310

COMMON NEPMAX	G4 09320
COMMON N , M , A , B	G4 09330
1 , R , N2 , N1 , NR	G4 09340
2 , NM , KFLAG , JFLAG , COST	G4 09350
3 , LEVM , NRN2 , NM1 , NN2	G4 09360
COMMON ISUCC(40,40) , LISUCC(40) , IPRED(40,40) , LIPRED(40)	G4 09370
1 , INC\$MX(40,40) , SUC\$MX(40,40) , P\$(2,1280) , UNAME(40)	G4 09380
2 , GLEVEL(40) , LGLIST(40) , HLIST(40,40) , TIME	G4 09390
COMMON T , RTCONN(100) , S , RSCONN(100)	G4 09400
COMMON IFLAG , POINTA , ESSIS(40) , F\$1(32)	G4 09410
1 , F\$UB1 , INPTCV(32) , LISTC(40) , POINTC	G4 09420



2	,LISTL(40)	,POINTL	,ORIGIN(40)	,IPATH(40)	G4 09430
3	,POINTR	,VF\$1(32)	,VF\$UB1	,GSMALL(40,32)	G4 09440
	COMMON POTAB(200,42),PPOTAB(40)		,LPOTAB(40)	,NRPLC(2)	G4 09450
1	,RPLC(2,40)	,IDX0(32)	,IDX0E(32)	,IDX1(32)	G4 09460
2	,IDX1E(32)	,SUMP(32)	,SETT1(32)	,NOT1	G4 09470
3	,SET\$1(40)	,NOS1	,SET\$ (40)	,NOS	G4 09480
4	,STS	,SUM\$2(32)	,SET\$2(200)	,NOS2	G4 09490
5	,LIP	,NDOE	,KEYA	,KEYB	G4 09500
6	,NOO	,NO1	,NO1E	, \$GT	G4 09510
7	, \$LTH	, \$PW	, \$NOE	,G\$ \$ \$ \$ \$	G4 09520
	COMMON NOT1SV	,NOS1SV	,LMT\$2		G4 09530
	DIMENSION X(40),LX(40,2),OUTO(40)				G4 09540
	ENTRY PRESUC				G4 09550
1	CONTINUE				G4 09560
	DO 10 GI=1,NR				G4 09570
	LS=0				G4 09580
	LP=0				G4 09590
	DO 5 GJ=1,NR				G4 09600
	IF(INC\$MX(GI,GJ).EQ.0) GO TO 3				G4 09610
	LS=LS+1				G4 09620
	ISUCC(LS,GI)=GJ				G4 09630
	GO TO 5				G4 09640
3	IF(INC\$MX(GJ,GI).EQ.0) GO TO 5				G4 09650
	LP=LP+1				G4 09660
	IPRED(LP,GI)=GJ				G4 09670
5	CONTINUE				G4 09680
	LISUCC(GI)=LS				G4 09690
	LIPRED(GI)=LP				G4 09700
10	CONTINUE				G4 09710
	ENTRY SUCCES				G4 09720
	DO 21 GI=1,NR				G4 09730
	DO 21 GJ=1,NR				G4 09740
	SUC\$MX(GI,GJ)=0				G4 09750
21	CONTINUE				G4 09760
	DO 30 GJ=N1,NR				G4 09770
	DO 22 GS=1,NR				G4 09780
	X(GS)=0				G4 09790
22	CONTINUE				G4 09800
	X(GJ)=1				G4 09810
	L0=1				G4 09820
	LX(1,1)=GJ				G4 09830
	V=1				G4 09840
23	CONTINUE				G4 09850
	V=1-V				G4 09860
	SW0=1+V				G4 09870
	SW1=2-V				G4 09880
	L1=0				G4 09890
	DO 28 LL=1,L0				G4 09900
	GM=LX(LL,SW0)				G4 09910
	LIP=LIPRED(GM)				G4 09920
	IF(LIP.EQ.0) GO TO 28				G4 09930
	DO 26 LP=1,LIP				G4 09940
	GP=IPRED(LP,GM)				G4 09950
	IF(X(GP).GT.0) GO TO 26				G4 09960
	SUC\$MX(GP,GJ)=1				G4 09970
	L1=L1+1				G4 09980
	LX(L1,SW1)=GP				G4 09990
	X(GP)=1				G4 10000
26	CONTINUE				G4 10010
28	CONTINUE				G4 10020
					G4 10030



IF(L1.EQ.0) GO TO 30	G4 10040
L0=L1	G4 10050
GO TO 23	G4 10060
30 CONTINUE	G4 10070
ENTRY LEVEL	G4 10080
DO 40 GJ=1,NR	G4 10090
OUTO(GJ)=LISUCC(GJ)	G4 10100
GLEVEL(GJ)=-1	G4 10110
40 CONTINUE	G4 10120
LEV=0	G4 10130
45 LEV=LEV+1	G4 10140
G=0	G4 10150
DO 50 GJ=1,NR	G4 10160
IF(OUTO(GJ).GT.0 .OR. GLEVEL(GJ).GT.0) GO TO 50	G4 10170
G=G+1	G4 10180
HLIST(G,LEV)=GJ	G4 10190
GLEVEL(GJ)=LEV	G4 10200
50 CONTINUE	G4 10210
IF(G.EQ.0) RETURN	G4 10220
LGLIST(LEV)=G	G4 10230
DO 60 GG=1,G	G4 10240
GJ=HLIST(GG,LEV)	G4 10250
LIP=LIPRED(GJ)	G4 10260
IF(LIP.EQ.0) GO TO 60	G4 10270
DO 55 LP=1,LIP	G4 10280
GP=IPRED(LP,GJ)	G4 10290
OUTO(GP)=OUTO(GP)-1	G4 10300
55 CONTINUE	G4 10310
60 CONTINUE	G4 10320
LEVM=LEV	G4 10330
GO TO 45	G4 10340
ENTRY PVALUE	G4 10350
DO 100 L=NV2,NRN2	G4 10360
P\$(1,L)=1	G4 10370
100 CONTINUE	G4 10380
LEV=LEVM	G4 10390
110 CONTINUE	G4 10400
L0=LGLIST(LEV)	G4 10410
DO 130 L=1,L0	G4 10420
GI=HLIST(L,LEV)	G4 10430
LIS=LISUCC(GI)	G4 10440
BSGI=(GI-1)*N2	G4 10450
LJTH=0	G4 10460
DO 115 JTH=1,N2	G4 10470
IF(P\$(1,BSGI+JTH).EQ.0) GO TO 115	G4 10480
LJTH=LJTH+1	G4 10490
X(LJTH)=JTH	G4 10500
115 CONTINUE	G4 10510
IF(LJTH.EQ.0) GO TO 130	G4 10520
DO 125 LS=1,LIS	G4 10530
GS=ISUCC(LS,GI)	G4 10540
BSGS=(GS-1)*N2	G4 10550
DO 120 LJ=1,LJTH	G4 10560
P\$(1,X(LJ)+BSGS)=0	G4 10570
120 CONTINUE	G4 10580
125 CONTINUE	G4 10590
	G4 10600
	G4 10610
	G4 10620
	G4 10630
	G4 10640

130	CONTINUE	G4	10650
	LEV=LEV-1	G4	10660
	IF(LEV.GE.2) GO TO 110	G4	10670
	RETURN	G4	10680
		G4	10690
		G4	10700
	ENTRY RSTCT(KEYRST)	G4	10710
	KEYRST=0	G4	10720
	IF(LEVM.GT.LMAX)GO TO 160	G4	10730
	DO 150 GI=N1,NP	G4	10740
	IF(LIPRED(GI).GT.FANIN)GO TO 160	G4	10750
	IF(LISUCC(GI).GT.FANOUT)GO TO 160	G4	10760
150	CONTINUE	G4	10770
	RETURN	G4	10780
160	KEYRST=1	G4	10790
	RETURN	G4	10800
	ENTRY UNNECE	G4	10810
*****	THIS ENTRY DISCONNECT ALL GATES FROM WHICH THERE IS NO PATH	G4	10820
	TO OUTPUT GATES *****	G4	10830
	TS=T	G4	10840
	DO 209 GI=NM1,NP	G4	10850
	IF(GLEVEL(GI).EQ.1) GO TO 207	G4	10860
	DO 205 GJ=N1,NM	G4	10870
	IF(SUC\$MX(GI,GJ).GT.0) GO TO 209	G4	10880
205	CONTINUE	G4	10890
*****	GI IS REDUNDANT *****	G4	10900
207	CONTINUE	G4	10910
	LIP=LIPRED(GI)	G4	10920
	IF(LIP.EQ.0) GO TO 206	G4	10930
	DO 203 LI=1,LIP	G4	10940
	GK=IPRED(LI,GI)	G4	10950
	IF(INC\$MX(GK,GI).LE.0) GO TO 203	G4	10960
	T=T+1	G4	10970
	RTCONN(T)=100*GK+GI	G4	10980
	INC\$MX(GK,GI)=0	G4	10990
203	CONTINUE	G4	11000
206	LIS=LISUCC(GI)	G4	11010
	IF(LIS.EQ.0) GO TO 209	G4	11020
	DO 204 LI=1,LIS	G4	11030
	GK=ISUCC(LI,GI)	G4	11040
	IF(INC\$MX(GI,GK).LE.0) GO TO 204	G4	11050
	T=T+1	G4	11060
	RTCONN(T)=100*GI+GK	G4	11070
	INC\$MX(GI,GK)=0	G4	11080
204	CONTINUE	G4	11090
209	CONTINUE	G4	11100
	IF(T.GT.TS) GO TO 1	G4	11110
	RETURN	G4	11120
	END	G4	11130
		G4	11140



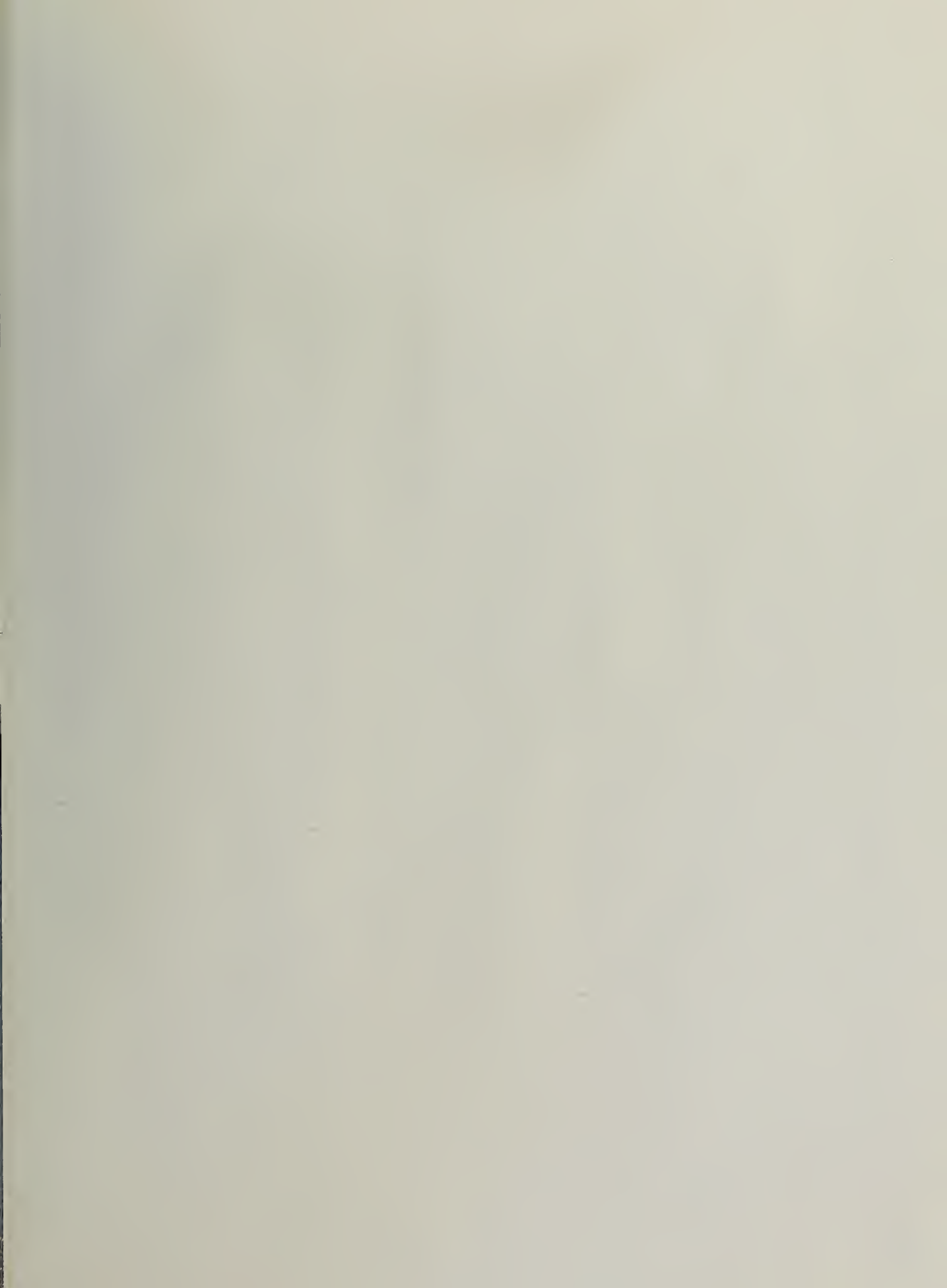
<b>BIBLIOGRAPHIC DATA SHEET</b>	1. Report No. UIUCDCS-R-75-714	2.	3. Recipient's Accession No.
4. Title and Subtitle PROGRAM MANUAL: NOR NETWORK TRANSDUCTION BY GERERALIZED GATE MERGING AND SUBSTITUTION (Reference Manual of NOR Network Transduction Programs NETTRA-G3 and NETTRA-G4)		5. Report Date April 1975	
7. Author(s) H.C. LAI		8. Performing Organization Rept. No.	
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. Project/Task/Work Unit No.	
		11. Contract/Grant No. NSF GJ-40221	
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C. 20550		13. Type of Report & Period Covered Technical	
		14.	
15. Supplementary Notes			
16. Abstracts This is a reference manual for NOR network transduction programs NETTRA-G3 and NETTRA-G4. NETTRA-G3 reduces the number of gates in a given network by means of merging of gates whereas NETTRA-G4 reduces the number of gates by means of substitution for all output connections of a selected gate. The principles of these programs will be discussed in more detail in another paper by the author and Y. Kambayashi.			
17. Key Words and Document Analysis. 17a. Descriptors Logic design, logic circuits, logical elements, programs (computers).			
17b. Identifiers/Open-Ended Terms Program manual, computer-aided-design, permissible functions, network transduction, network tranformation, gate merging, gate substitution, NETTRA-G3, NETTRA-G4, NOR, CSPP.			
17c. COSATI Field/Group			
18. Availability Statement Release Unlimited		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 107
		20. Security Class (This Page) UNCLASSIFIED	22. Price





218 6 PM













OCT 13 1977



UNIVERSITY OF ILLINOIS-URBANA  
510.84 IL6R no. C002 no. 713-714(1975  
Internal report /



3 0112 088401952